

# Classifying Music Notes Into Genres:

## SIADS 693 Milestone II

By: Patrick Thornton, Pu Zeng, Braedon Shick

GitHub Repository: <https://github.com/Pu-Zeng/696-Milestone>

### I. INTRODUCTION

Paulo Coelho, a famous lyricist, once said, "The music speaks for itself. There's no need to say anything", which is the inspiration for our project. In this project, we want to classify and predict the genre, the emotion of a particular piano track from the music notes, and detect the topics of lyrics with unsupervised learning. The project can help generate tags for new music automatically, and more importantly, we can explore something deeper in the nature of music with the model results. Is there a relationship between music notes, genre, and the emotions conveyed?

CNN and LSTM have an attention layer used for the genre classification and emotion detection tasks. We also use a prompt fine-tuned Flan-T5 model to detect lyric emotions to generate emotion labels for the Musical Instrument Digital Interface (MIDI) music. Both our CNN and LSTM models have a 66% accuracy for the 17 genre classification task. This hints that music genres have deep roots in their notes. However, the emotion detection result is subtle. Our CNN model can, at best, achieve an accuracy of 55% for 11 emotions, which is better than a random classifier but not impressive enough. We used the same model structure for an arousal and valence score regression task with MFCC features of songs and achieved a R-squared score of 0.4. Therefore, perhaps musical notes can provide some foundation for music's emotions, but the singer's voice and the musician's play give it an exclusive soul.

Lyrics are another important part of the music and are used in our unsupervised learning section to find clusters. As lyrics are informal and short text, we use the TC-W2V score rather than the traditional coherence score to find the optimal number of topics and evaluate our NMF and LDA models. It's interesting to find that the optimal numbers of clusters are quite similar to both NMF and LDA models and not far from the number of emotions generated by the Flan-T5 model in our supervised learning section.

### II. RELATED WORK

**Example 1:** An example of MIDI classification in GitHub is the MIDI Classification Tutorial on a different dataset ([example](#)), which classifies the music by its statistical information with multilayer perceptron and support vector machine. However, this tutorial only results in 13 main categories of music and ignores the unbalanced nature of the Lakh-Matched dataset. We are improving upon this example by using resampling to form a balanced dataset that has better represented genres.

**Example 2:** Another inspiring and well documented example of music genre classification is an article where [Raghav Argawal](#) used k-means clustering to identify music genre from wav audio files. This thorough article used MFCC to extract features focused on the amplitude of frequencies from the audio files. After applying k-means clustering the model achieved 70% accuracy on 10 different classes. The dataset we are using stores individual songs as an MIDI file, which is less informative than a wav audio file, and achieves a similar accuracy on 17 classes.

**Example 3:** A well documented music genre classification project was created by [Param Ravel](#), where the dataset consisted of 1,000 song samples, all 30 seconds long. The project explored classification using the LSTM model where the model accuracy was 80% on 10 classes through a very low learning rate, 60 epochs, and 85% validation accuracy. We use LSTM for 17 genre classification on MIDI files, which is less informative than WAV, and achieve an accuracy of 66%.

### III. DATA SOURCES

In Milestone 2, we spent most of our time trying various datasets and models to reach our goals. The datasets used in our final models are listed below.

Data Set	Format	Location	Used for	Records	Notes
ADL Piano[1]	MIDI	<a href="https://github.com/lucasnfe/adl-piano-MIDI">https://github.com/lucasnfe/adl-piano-MIDI</a>	Supervised	11,086	MIDI file with only piano instruments.

Lakh-Matched[2][3]	MIDI	<a href="https://colinraffel.com/projects/lmd/">https://colinraffel.com/projects/lmd/</a>	Supervised	45,129	MIDI files, and 1810 of them have embedded lyrics.
Emotion in Music Database[4]	CSV, MP3	<a href="https://cvml.unige.ch/databases/emoMusic/">https://cvml.unige.ch/databases/emoMusic/</a>	Supervised	1,000	Well annotated arousal and valence scores of each music.
VGMIDI Dataset[5]	MIDI, JSON	<a href="https://github.com/lucasnfe/vgMIDI">https://github.com/lucasnfe/vgMIDI</a>	Supervised	200	200 MIDI files were annotated by 30 human subjects with arousal and valence score.
The musixmatch Dataset Training set	TXT	<a href="http://millionsongdataset.com/musixmatch/">http://millionsongdataset.com/musixmatch/</a>	Unsupervised	210,519	The official collection of Million Song dataset lyrics in bag of word format.
Spotify Audio Features	CSV	<a href="https://www.kaggle.com/datasets/imuhammad/audio-features-and-lyrics-of-spotify-songs/">https://www.kaggle.com/datasets/imuhammad/audio-features-and-lyrics-of-spotify-songs/</a>	Supervised	18,000	Contains audio features and lyrics

Table 1 Datasets

The **ADL Piano dataset** consists of 11,086 piano pieces from different genres. We use this dataset because it provides MIDI files with only piano instruments, which we found to be the most representative of genre classification in our experiments.

The **Lakh-Matched** dataset is four times larger than ADL Piano. We needed this dataset because only a small portion of MIDI files in the ADL Piano dataset have embedded lyrics, so we need to expand the size of our dataset for emotion detection.

The **Emotion in Music Database** consists of 1,000 MP3 clips with well-annotated arousal and valence scores. Each clip is annotated by a minimum of 10 workers, ensuring the dataset's accuracy.

The **VGMIDI Dataset** consists of 200 video game MIDI files with well-annotated arousal and valence scores. We use this dataset because we believe its annotation helped our model to capture some subtle relationship between emotion and music notes, and other datasets we found are not as well annotated as this one.

The **Spotify Audio Features** dataset contains features of 18,000 songs in Spotify's library. Some of these features are high-level, like name, artist, album, or genre (represented by playlist genre), while others are more descriptive, such as the loudness, key, or tempo. We are using this dataset to set a baseline given its readiness to be trained on. The features we used from this dataset are already numerical, and the target class is balanced.

The **musixmatch Dataset Training set** provides over 210,000 music lyrics, which is a large dataset for our unsupervised lyric machine learning.

## IV. FEATURE ENGINEERING

### 1. GENRE CLASSIFICATION

#### 1). MIDI to Matrix

MIDI data is symbolic music notes, which is different from the audio formats we are more familiar with, such as MP3 and WAV files. We can find different messages controlling when to press and release a key on the piano in MIDI files and its corresponding music sheets like the picture below.

## Milestone II - braedon-patrikt-puzeng

```
MidiTrack([
  Message('program_change', channel=0, program=0, time=0),
  MetaMessage('key_signature', key='C', time=0),
  Message('note_on', channel=0, note=69, velocity=114, time=0),
  Message('note_off', channel=0, note=69, velocity=114, time=120),
  Message('note_on', channel=0, note=72, velocity=114, time=0),
  Message('note_off', channel=0, note=72, velocity=114, time=120),
  Message('note_on', channel=0, note=74, velocity=114, time=0),
  Message('note_off', channel=0, note=74, velocity=114, time=120),
  Message('note_on', channel=0, note=76, velocity=114, time=0),
  Message('note_off', channel=0, note=76, velocity=114, time=120),
  Message('note_on', channel=0, note=69, velocity=93, time=0),
  Message('note_off', channel=0, note=69, velocity=93, time=120),
  Message('note_on', channel=0, note=72, velocity=93, time=0),
  Message('note_off', channel=0, note=72, velocity=93, time=120),
  Message('note_on', channel=0, note=74, velocity=93, time=0),
  Message('note_off', channel=0, note=74, velocity=93, time=120),
  Message('note_on', channel=0, note=76, velocity=93, time=0),
  Message('note_off', channel=0, note=76, velocity=93, time=120),
  Message('note_on', channel=0, note=69, velocity=104, time=0),
  Message('note_off', channel=0, note=69, velocity=104, time=120),
  Message('note_on', channel=0, note=72, velocity=104, time=0),
  Message('note_off', channel=0, note=72, velocity=104, time=120),
  Message('note_on', channel=0, note=74, velocity=104, time=0),
  Message('note_off', channel=0, note=74, velocity=104, time=120),
])
```



Figure 1 MIDI file and music notes

Therefore, we need to translate the music notes into matrices that can be efficiently processed by a computer. Some MIDI libraries such as `pretty_MIDI`, `mido`, and `pypianoroll` provide powerful algorithms to transform MIDI music into matrices, where each row corresponds to a time frame, and each column corresponds to a music key.

However, because of the different tempos and resolutions of the MIDI files, the transformed matrices are not time equivalent. For example, ten rows in one matrix may represent 1 second, while ten rows in another matrix represent 0.5 seconds in another matrix. Another problem is caused by the variety of the music length, which results in different numbers of rows of matrices. What's more, one MIDI file always contains thousands of rows, which may be too large to process, so we needed to find a way to compress them.

To solve the problem, we scaled each matrix so that each row lasts 0.1 seconds with the height calculated by the formula below.

$$Height = \frac{Tempo}{Resolution \times 60 \times 0.1}$$

After the transformation, each row of each matrix corresponds to the same time period, and we chose frames of a fixed length, for example, 512 frames, which correspond to 51.2 seconds of music as the dataset. This data can be processed efficiently by a CNN or LSTM network as they have the same size.

Another problem we needed to solve comes from each MIDI file recording music notes from different instruments. Though it's possible to store each instrument as different channels of the data matrix - such as Acoustic Grand Piano in channel 0 and Clavinet in channel 7, it will be memory-consuming and cause a lot of burden. As the piano often plays the main melody of music, we used the mean intensity of all the piano instruments as the value in the matrix.

## 2). Generate the Labels

ADL Piano dataset organized the MIDI files into category directories. However, each directory contains many subdirectories which correspond to subgenres. We use a DFS algorithm to traverse all the files and use the root directory name as the genre label.

## 2. EMOTION DETECTION

### 1). MIDI Lyric to Emotion Labels

Though there is an official bag of word representation of lyrics for the Lakh-Matched dataset, we found the average Word2Vec embedding is not an accurate representation of music emotions as there are some harmful or offensive topics extracted and are sometimes too general. Perhaps sequential lyrics can provide more information on excellent emotion labels.

Luckily, some MIDI files have embedded sequential lyrics. We needed more MIDI files to form a reasonable dataset size, so we scanned all the MIDI files in the Lakh-Matched dataset and found 1,810 of them. We deleted all the lyric lines starting at exactly 0:00 of each MIDI file as they are more likely to have copyrighted information and have no relation to emotion.

## Milestone II - braedon-patrikt-puzeng

As mentioned above, classical average Word2Vec embeddings are more likely to generate inappropriate words or generic labels, so other solutions need to be found for label extraction. As large language models are famous for their performance on different NLP tasks, we use the Flan-T5-Base model to summarize lyrics to an emotional word. We first used prompt tuning to make the pre-trained Flan-T5-Base model generate only one word and used the lyrics of each song as input to get the labels. Some labels generated by Flan-T5 are not representative enough, or might return a sentence even though we asked for single word representations, so we deleted all these labels and formed a dataset of 1,143 songs.

We applied the same technique as what we did for the ADL Piano dataset to extract music matrices.

## 2). Audio Features

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound based on a linear cosine transform. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC and are derived from a type of cepstral representation of the audio clip.

We transform MP3 files directly into MFCCs with Librosa Library. For MIDI files, we first transform them into WAV files with FluidSynth and then use Librosa to transform them into MFCCs. In our projects, we chose the number of MFCCs to be 20, which is commonly used in the audio processing field.

## 3. UNSUPERVISED LEARNING

The official Million Song lyric dataset uses the bag of words data structure to represent lyrics, and we use TF-IDF representation for later NMF and LDA models. Therefore, we can either use the TF-IDF formula or apply the Scikit-Learn vectorizer on the joined string of the lyric word list, where each word is replicated by its frequency in the particular lyric. We use the latter method in our model as it works better with later Scikit-Learn models.

## 4. FINAL FEATURE LIST

The final features we used with Spotify Dataset is shown in detail in Appendix III

Feature Name	Format
Spotify Features	Double
MIDI Notes Matrix	512 (time frames) * 128 (keys)
MIDI MFCC Matrix	Time Frames * 20
MP3 MFCC Matrix	Time Frames * 20
Word Embeddings	50 Dimensions

Table 2 Data structures

## V. SUPERVISED LEARNING

Our supervised learning work consists of 4 parts - SVM, Random Forests, and Logistic Regression on the Spotify dataset as the baseline models for comparison, CNN and LSTM on the ADL Piano dataset for the genre classification task, CNN and Flan-T5 generated labels for emotion classification task, and CNN and LSTM on MFCCs and music notes from audio and MIDI file to regress the arousal and valence score.

### 1. GENRE CLASSIFICATION BY OUT-OF-BOX SCIKIT-LEARN MODELS

#### A. Structure

To get an understanding of what sort of accuracy we can expect from a basic classifier, we utilized some of scikit-learn's simpler models, trained on a dataset more compatible with these types of models. The Spotify Audio Features dataset contains information on 18,000+ songs, including the playlist genre, which will be the target variable of our model. The genre of a song in this dataset can take one of six values: Rock, R&B, Pop, EDM, Latin, and Rap. Outside of the target variable, there are several numerical features in the dataset that we are able to train our models on without much need for feature engineering. During pre-processing, all that was needed was to select features that were not specific to a song (e.g., song name or artist correlated

## Milestone II - braedon-patrikt-puzeng

highly with the genre), which essentially left us with only the numerical ones, and scaling them using sklearn's StandardScalar class.

The details of features can be found in Appendix III.

## B. Experiments

Once the data was prepared, we wanted to test out the different classifiers to see if any performed considerably better than the others to give this simple approach a fair chance. To do this, we trained seven separate models: Dummy Classifier (DC), Naive Bayes (NB), Stochastic Gradient Descent (SGD), Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boost (GB), and Random Forest (RF). We compared the different models using the results of a cross-validated score. Outside of the DC, the result showed that while some models outperformed others, none were exceptionally better than the rest.

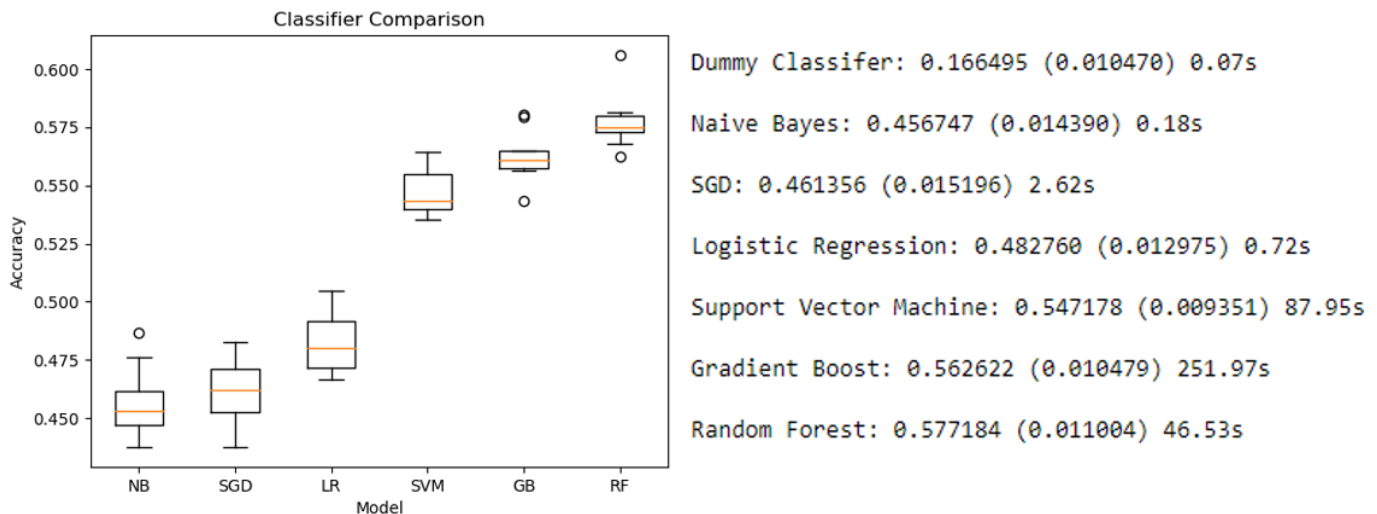
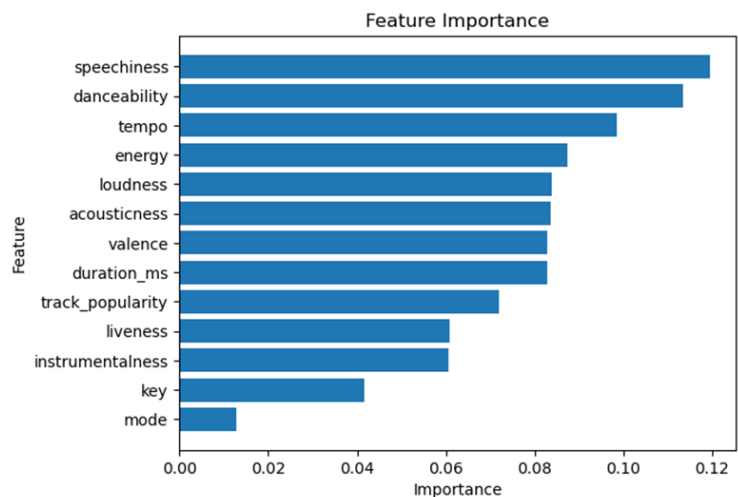


Figure 2 Model accuracy

Among these model results, we can see that our random forest model not only performed the best but executed considerably quicker than the other two models of similar performance. With our model type selected, we wanted to see how impactful the different features were and if there was anything we should remove before the lengthy process of hyper-parameter tuning. To do this, we retrained a random forest classifier and printed out its feature importance. We found that while each feature did contribute to the accuracy, *mode* had little impact. To decide if we wanted to remove this feature, we did a quick check on its impact on runtime and accuracy.

We saw that removing *mode* did improve runtime at the cost of accuracy, but its impact on the runtime was not significant enough to remove it from the training set. To verify that we had the best model we could get using this dataset, we utilized the scikit-learn GridSearchCV class to test out a slew of different hyper-parameter settings.



Accuracy and train time with mode: 0.582, 3.97s  
 Accuracy and train time without mode: 0.585, 4.22s

Figure 3 Feature importance

## C. Evaluation

### C.1: Sensitivity Analysis

When gauging a model's sensitivity to new data, there are a few indicators we can look at without having access to actual new data. Two of these indicators are an overreliance on specific features or oversensitivity to different hyperparameter settings.

## Milestone II - braedon-patrikt-puzeng

During our grid search, we tested 225 different hyperparameter setting combinations, and almost all results scored above 0.49 accuracy. This consistency indicates that the model is resilient in the face of different hyperparameter values. As for the over-reliance on specific features, in our exploration of the importance of the features, we saw that no single feature scored particularly high in importance. This indicates that if our model were introduced to new data that had significantly different values for any given feature, we wouldn't expect that to lead to equally different results.

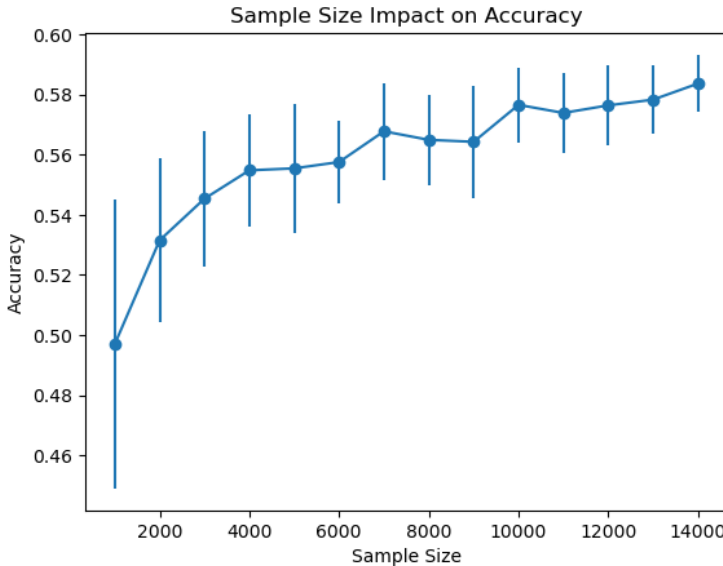


Figure 4 Sensitivity Analysis

Lastly, we verified that the model would not be significantly different in future iterations due to an increase in the size of the training dataset. This is proven by testing out the accuracy of our model using different sample sizes of our training dataset. In our tests, we found that after around 2,000 records, the model's accuracy only improved in small increments as we added additional records 1,000 at a time. What these results tell us is that while new data could be introduced in the future, and the model could be retrained on more data, the outcome would not drastically change.

Through these different tests, we are confident in saying that our model is very resilient to changes in various solution elements.

### C.2: Trade Offs

In the training and optimization phases of this model, we had to make several decisions in terms of model choice, feature selection, hyper-parameter settings, and scoring choice. The decisions on these tradeoffs were always aimed at balancing accuracy and run time.

Of the models, we had a few comparable options between SVM, Gradient Boost, and Randoms Forest. While it would have been ideal to test out all three models after feature selection and hyper-parameter tuning, the former two options took twice and four times as long, respectively, as the RF. In this case, we made the tradeoff of comprehensive testing to achieve results in a realistic timeframe.

Between feature selection and hyper-parameter settings, there were not any tough choices. The removal of the one feature that had little impact on the outcome did not reduce runtime significantly, so it was left in. Hyper-parameter tuning was done through brute force, with little impact on total training time. We forewent worrying about runtime and took the most accurate results.

Our last decision had to be made about the scoring method. In many applications of machine learning, disproportionate importance is placed on making sure the model does not accidentally predict a specific class (precision) or that the model doesn't miss a specific class (recall). In our experiment, however, we only care about how often the model correctly classifies the song genre. To calculate how often the model was correct, we used the model's accuracy, which divided the total correct predictions by the total number of predictions.

### C.3: Final results

Our results showed that the best settings of the options we provided were as follows: 'criterion': 'gini', 'max\_depth': 50, 'max\_features': 'auto', 'n\_estimators': 500. Using these settings, a random forest model trained on a dataset consisting of audio features alone could score at highest a 0.59.

## 2. GENRE CLASSIFICATION BY NEURAL NETWORKS

After transformation, the MIDI music can be represented as a matrix, where each row corresponds to 0.1 seconds of the music, and each column corresponds to a particular key. The matrix's value is the key's intensity at a specific time. It's reasonable that multiple keys are played together, so multiple non-zero values on the same row are allowed.

## Milestone II - braedon-patrikt-puzeng

This data representation can be considered similar to a one-channel image, where the time frames are aligned on the y-axis from the top to the bottom, and keys are aligned on the x-axis from low to high. Therefore, it's natural to use some regular image classification algorithms such as CNN for this task. As a 1-dimensional CNN works like an n-gram model, which is similar to how LSTM handles the data, we use a two-dimensional CNN to avoid models with a repetitive nature.

The matrix can also be considered as sequential data, where rows imply the order and columns work like the features of that particular time. From this perspective, we believe LSTM is a promising candidate for the classification task.

The structure of our CNN model can be found in Appendix IV.

### 1) CNN Classification

#### A. Structure

Our CNN model consists of five convolutional and max pooling layers, followed by a fully connected layer to output each genre's predicted probability. Both the depth of the CNN and the sizes of convolution and pooling kernels are restricted by the data matrix size.

Different from regular image classification problems, where a matrix's length and height are similar, our project's matrices are much narrower. This phenomenon is caused by the number of keys being only 128, while the number of time frames is much larger (512 here). It's difficult to select a smaller sampling rate to shorten the height and keep the information for some music notes. For example, if we use a sample rate larger than 0.125s, the information of eighth notes will be lost on 60 beats per minute condition. Therefore, the kernel size of a pooling layer needed to be narrowed.

The matrix and kernel sizes also affect the depth of the CNN. After each convolutional layer and pooling layer, the hidden layer outputs a much smaller matrix. If we use too many layers, the output size will quickly become zero and cause an error.

Therefore, the depth of the CNN is dependent on kernel sizes if the input matrix size is fixed. We set the input matrix to have 512 rows, as most of our MIDI files last longer than 51.2s, so we can keep as many of these files as possible in our dataset. Then we tried different kernel sizes, 2\*1, 2\*2, 5\*1, 5\*2, 10\*1, and 10\*2 for the convolutional layers and 2\*1, 3\*1 for the pooling layers. Finally, we find that a combination of 10\*1 convolutional kernels for the early layers, 5\*2 for later layers, and a pooling kernel size of 2\*1 works better. The convolutional size is reasonable considering the nature of the music, as the receptive field of the first convolutional layers is half or one beat for MIDI files with 60 or 120 beats per minute.

#### B. Experiments

The original dataset is unbalanced among different genres. Therefore, we first split the dataset into two datasets - a training & validation set and a test set to avoid data leakage problems. In each cross-validation process, we split the training & validation set into the training set and validation set to make sure there is no overlap between them, and then resample each dataset with replacement to make them balanced. At the test stage, we resample the test set with replacement to make it balanced.

We experimented with different sets of features for this CNN. The first is the average intensity matrix of all the available instruments in a particular MIDI file, the second is the average intensity of all the piano instruments, and the third is the MFCC matrix with all instruments of that MIDI file. It's interesting to find that the matrix of all instruments and MFCCs works only slightly better than a random guess, while the CNN on piano matrix achieves an accuracy of nearly 65%.

Perhaps only the main melody in music contains most of the information to identify a music genre, and the average of all the instruments makes the main instrument obscure. As the piano is often chosen as the instrument for the main melody, its intensity resulted in better accuracy for genre classification.

Using a more complex matrix with multiple channels rather than one channel, where each channel corresponds to a particular instrument instead of the average, might improve accuracy. The

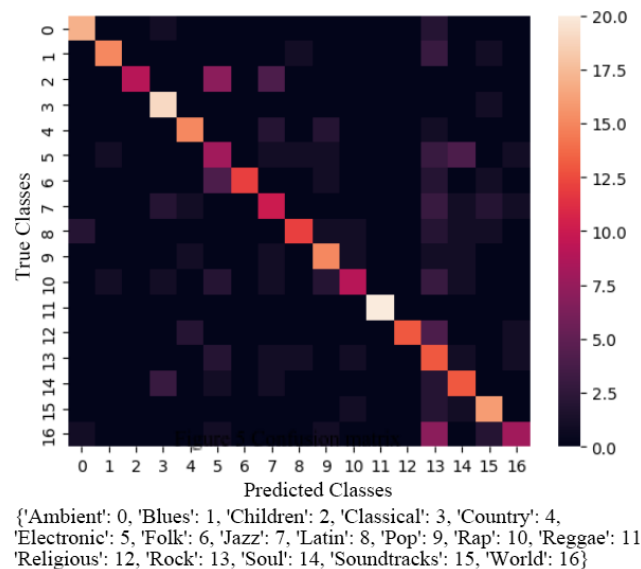


Figure 5 Confusion matrix of CNN genre classification model



## Milestone II - braedon-patrikt-puzeng

	Precision	Recall	F1	Support
Ambient	0.85	0.85	0.85	20
Blues	0.75	0.88	0.81	17
Children	0.45	1.00	0.62	9
Classical	0.95	0.73	0.83	26
Country	0.75	0.79	0.77	19
Electronic	0.40	0.32	0.36	25
Folk	0.60	1.00	0.75	12
Jazz	0.50	0.48	0.49	21
Latin	0.60	0.80	0.69	15
Pop	0.75	0.65	0.70	23
Rap	0.45	0.69	0.55	13
Reggae	1.00	1.00	1.00	20
Religious	0.65	1.00	0.79	13
Rock	0.65	0.27	0.38	48
Soul	0.65	0.57	0.60	23
Soundtracks	0.80	0.67	0.73	24
World	0.40	0.67	0.50	12
Average	0.66	0.73	0.67	20

Table 3 Evaluation score of CNN genre classification model

- to have some clear characteristics and are more likely to be identified, while some others are more confusing.
- Interestingly, our model has much higher recalls on Children, Folk, Latin, Religious, and World music than their precisions. Perhaps these music genres have some shared patterns with other genres in piano music notes and confuse our model. For example, our model tends to misclassify children's music with folk music and Latin music, as these genres are likely to have simple melodies in a storytelling style.
- It's very interesting to find that the model performed well on soundtracks in both precision and recall evaluation, considering their few limits and the diversity of movie themes.

## 2) RNN Classification

### A. Structure

We used the same 512\*128 matrices as the input of our CNN above and tried models with 1, 2, 3, 50, and 100 layers of LSTM. Using 2 LSTM layers is a good balance of model performance and computation consumption.

Beyond the basic LSTM structure, bidirectional LSTM increases the accuracy from 55% to 60%, and an attention layer[6] further increases the accuracy to 66.50% with the piano dataset.

The performance of the LSTM neural network on different data structures is the same as the CNN network - the piano data worked far better than the two other data representations.

### B. Experiments

We used the same train-test splits and 5-fold validation method as the CNN experiments. The average accuracy of the five selected

corresponding channel will be all zeros if an instrument is not used in the music. However, it will take 80 gigabytes and be quite memory-expensive, so we leave this approach for future explorations.

### C. Evaluation

We used train-test splits and 5-fold validation methods to form the training, validation, and test set. Validation sets were used to find the best models, and the test set was used to evaluate the result. The average accuracy of the five selected models on the test set is 59.9% with 17 classes.

We then assembled the five selected models to calculate the average softmax results for the classification task and achieved a 65.90% accuracy. The confusion matrix and evaluation scores of the assembled model are listed.

Our model has a good performance on Ambient, Blues, Classical, Country, Folk, Religious, and Soundtrack music while finding it difficult to identify Electronic, Jazz, Rap, and Rock music. Instruments or human voices always play an important role in the music that confuses our model. For example, drums are important in Rock music, and human voices are the soul of Rap. Therefore, the problem is caused partly by our preprocessing of the MIDI data, where we only choose the piano music notes into the matrix. On the other hand, our model achieved high precision and recall scores for some music genres that have simple drum patterns, such as Reggae. Some interesting phenomena can be found in the evaluation scores.

- Our model has much higher precision on classical and rock music than their recalls. It seems a subset of these music genres tend

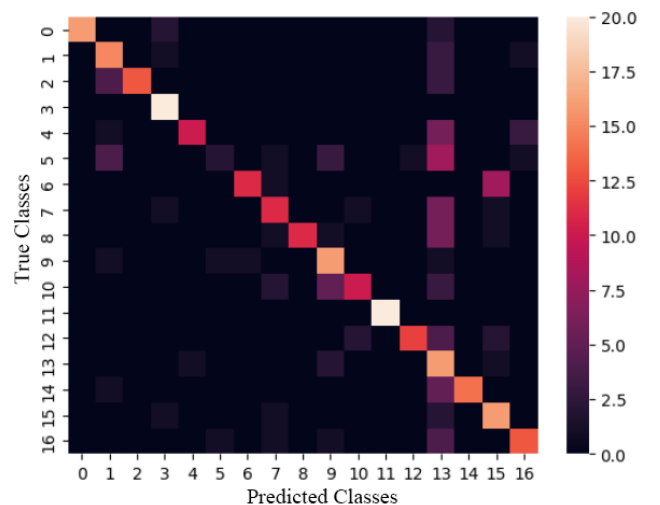


Figure 6 Confusion matrix of RNN genre classification model



## Milestone II - braedon-patrikt-puzeng

models on the test set is 61.8% on 17 classes, and the ensembled model achieved an accuracy of 66.50%. The confusion matrix of the assembled model and the evaluation scores are listed.

	Precision	Recall	F1	Support
Ambient	0.80	1.00	0.89	16
Blues	0.75	0.58	0.65	26
Children	0.65	1.00	0.79	13
Classical	1.00	0.80	0.89	25
Country	0.50	0.91	0.65	11
Electronic	0.10	0.50	0.17	4
Folk	0.55	0.92	0.69	12
Jazz	0.55	0.61	0.58	18
Latin	0.55	1.00	0.71	11
Pop	0.80	0.57	0.67	28
Rap	0.50	0.77	0.61	13
Reggae	1.00	1.00	1.00	20
Religious	0.60	0.92	0.73	13
Rock	0.80	0.23	0.36	69
Soul	0.70	1.00	0.82	14
Soundtracks	0.80	0.55	0.65	29
World	0.65	0.72	0.68	18
Average	0.66	0.77	0.68	20

Table 4 Evaluation score of RNN genre classification model

### 1) Semi-Supervised Learning Method

We can find 1,183 MIDI files with embedded English lyrics in the Lakh-Matched dataset and need to find a way to generate emotion labels from these lyrics. We chose Flan-T5 with prompt fine-tuning as our label generator. Each lyric set is inputted into the Flan-T5-Base model with two example prompts, and we choose the top 11 emotion labels and their MIDI files as our classification dataset.

The dataset is unbalanced with different labels because of the small number of MIDI files with lyrics. To compensate for the imbalance, we use the same resampling method as what we did in genre classification tasks..

### A. Classification Neural Network Structure

The model structure is similar to what we did in genre classification CNN models. We tried MIDI matrices and MFCCs from all instruments' intensity for the Lakh-Matched dataset, but the model performs no better than a random classifier. Then, we used the music matrix from the piano instruments and achieved an average accuracy of 44.27% with 5-fold validation on 11 classes.

One potential improvement we did not test out was using a language model other than Flan-T5-Base for label generation. Perhaps using a larger language model such as Chat-GPT may improve our results a lot, but it's a little expensive, and our goal in this iteration is just experimental.

### B. Experiments

We used the same train-test splits and 5-fold validation method as before. The average accuracy of the five selected models on the test set is 44.27% on 11 classes, and the ensembled model achieved an accuracy of 54.55%. The confusion matrix of the assembled model and the evaluation scores are listed below.

### C. Evaluation

The performance pattern of the RNN model is quite similar to our CNN model, and it seems to sacrifice some performance on precision to achieve a better recall.

a. We can find some differences in precision and recall in the RNN model compared with the CNN model. The recalls of Country and Electronic music are 7% and 12% higher in the RNN model, while the precisions decrease by 25% and 30%.

b. For some music genres, RNN performs better than the CNN model. For example, the Recalls increased by 43% and 5% for the Soul and World music with RNN, with an increase of precision by 5% and 25%.

c. On the other hand, CNN performs better in genres like Blues and Religious music.

Therefore, it seems if we use another XGboost to ensemble our CNN and RNN models, the results may improve. However, we haven't tried this solution due to the limited size of our dataset and time.

### 3. EMOTION DETECTION BY NEURAL NETWORKS

Emotion recognition from music is a more challenging task as we did not find good datasets with abundant annotated emotions. Therefore, we use semi-supervised learning methods and annotated arousal and valence score datasets to train our model.

### C. Evaluation

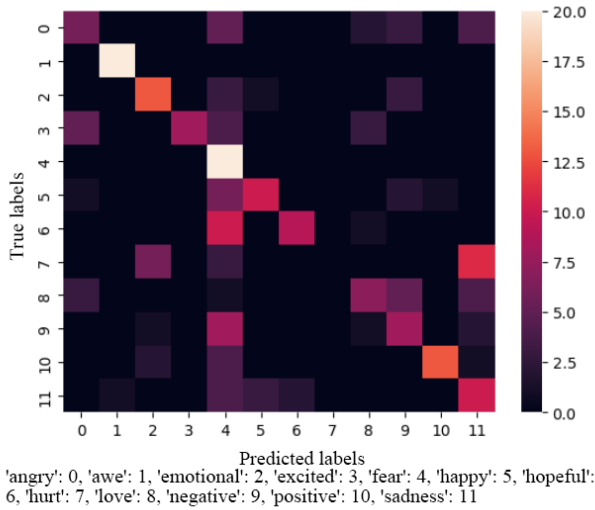


Figure 7 Confusion matrix of CNN emotion classification model

	Precision	Recall	F1	Support
Angry	0.55	0.85	0.67	13.00
Awe	1.00	1.00	1.00	20.00
Emotional	0.75	0.60	0.67	25.00
Excited	0.65	1.00	0.79	13.00
Fear	1.00	0.38	0.55	53.00
Happy	0.60	0.33	0.43	36.00
Hopeful	0.45	1.00	0.62	9.00
Hurt	0.00	0.00	0.00	0.00
Love	0.35	0.37	0.36	19.00
Negative	0.20	0.33	0.25	12.00
Positive	0.45	0.90	0.60	10.00
Average	0.55	0.61	0.54	19.09

Table 5 Evaluation score of CNN emotion classification model

This method is just experimental and shows some hope of using LLM for label generation.

Considering there are 11 classes, the accuracy is not great but better than a random guess, which means there are some patterns between emotions and music sheets. The CNN model may be more powerful as the emotions are only annotated by LLM rather than human beings.

We can find some patterns or insights from the evaluation matrix below.

- The model achieved a much higher average recall than precision, which means the model learned some patterns of music emotions, while these patterns may not be clear enough. For example, the model predicts no music for hopeful emotion while there are 20 pieces in the test set. Perhaps the Flan-T5-base model limited the quality of labels, and a better LLM model can alleviate this problem.
- Though the model failed to perform well on some of the emotions, it achieved excellent precision and recall scores on music labeled with awe and negative, which seems to be promising for further exploration with a larger dataset and a better LLM.

## 2) Arousal and Valence Regression with CNN network

### A. Model Structure

In this task, we found that the most important element to success is to find a suitable representation of the MIDI music. As we want the MIDI data structure to be compared with the MFCCs from the MP3 Audio files, we select the most intense instrument music sheet in a MIDI file and transform it into MFCCs.

We use the same CNN structure as before, only changing the size of the kernel size, the fully connected network, and the loss function for the regression task. For this experiment, a RNN model can also be used, but it takes significantly more time to train and requires larger memory.

### B. Experiments

Our experiment used two datasets to find the emotional contributions of different components, such as music notes, instrument tones, and human voices.

- MIDI files were transformed into WAV format with some standard sound fonts to include the tones and characteristics of instruments. The WAV files were then transformed into MFCC matrices so that the features from different datasets could be comparable. We also tried to use data structures like what we used in the classification task and tried different instrument tracks, but the result was not significant. It seems MFCCs work better for MIDI emotion detection.
- Audio data, which includes human voices and played by musicians, were directly transformed into MFCC matrices.

Milestone II - braedon-patrikt-puzeng

### C. Evaluations

Though we use the same structure on MFCCs from audio MP3 songs and MFCCs from MIDI tracks, we got a 0.4 R-squared on the audio data, while only 0.08 on the MIDI track. The key difference between these two music files is that audio data has the singer's voice and music played by musicians, while MIDI music is played exactly to the music notes with no skills from musicians and singers.

It seems that the composer provides some basis for different musical emotions, and it's the singer and musician who inject the soul into the final music.

## 4. TRADE-OFFS BETWEEN DIFFERENT DATASETS, RANDOM FOREST, AND NEURAL NETWORKS

Though neural networks achieved better accuracy, there are some trade-offs between these two families of models.

1) The Spotify dataset (41.9 Mb) is smaller than the ADL\_Piano dataset (109 Mb), as some of the most important features have been extracted from the original audios. In fact, the neural network transformed input are much larger (usually in Gbs) after pre-processing. Though we are not sure how much data Spotify used to extract these features, using features available in Spotify takes much less memory and preprocessing work if accuracy isn't the top 1 priority.

2) Random forest can be trained using far less computation power, usually well enough with CPU cores. In contrast, GPUs are needed for neural network training, especially for the LSTM model, which needs an A100 to be trained in an acceptable time.

## VI. UNSUPERVISED LEARNING

Our unsupervised learning work consists of 2 parts - LDA (Latent Dirichlet Allocation) and NMF (Non Negative Matrix Factorization) on a lyrical dataset. The LDA and NMF models use the Million Song official lyrical dataset to help decompose different lyrics into topics. This allows us to understand the common word categories used throughout the song industry.

### 1. TOPIC CLUSTER VISUALIZATION THROUGH AVERAGE EMBEDDINGS

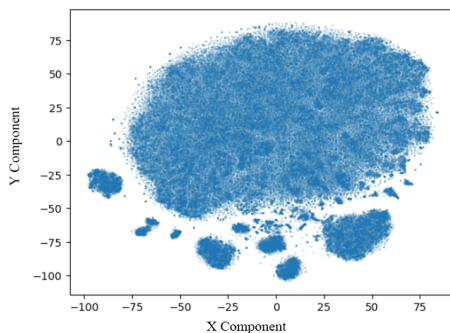


Figure 8 TSNE on average word embeddings word embedding space.

We use the Gensim glove-wiki-gigaword-50 pre-trained model to calculate the averages of Word2Vec embeddings for each lyric and use T-SNE methods for visualization. This may provide us with some hints about how many clusters to find.

The glove-wiki-gigaword-50 model is trained on aggregated global word co-occurrence statistics from the wiki dataset and output word embeddings, which represent some semantic meaning. TSNE is a dimension reduction method that focuses on preserving the local distance between neighbors and is a good candidate method for visualizing high-dimensional data.

We do not need to pre-define the number of clusters with word embeddings and TSNE methods, which can give us some hints about choosing hyper-parameters in our later LDA and NMF analysis.

From the results, we can find that there are almost 6-10 clusters in the average

### 2. TOPIC DETECTION THROUGH SKLEARN LATENT DIRICHLET ALLOCATION

#### A. Structure

We use TF-IDF to represent the lyrics to emphasize their uniqueness and use an LDA model for topic decomposition. LDA is a probabilistic method that learns to extract topics from generative processes and assumes a Dirichlet distribution of words to each topic and topics to each lyric. It outputs the probability of each topic belonging to a lyric. Therefore, we can use UMAP methods, which aim to preserve the local structure of the LDA output during the dimension reduction process, to visualize the LDA results. Compared with TSNE, UMAP preserves the local structure and some global distance at the same time.

#### B. Experiments

The most important hyperparameter for our LDA and NMF methods is the number of clusters. Though our work on the average word embedding of each lyric provides us with some hints, we still tried the number of clusters from 5 to 30 and compared their coherence scores.

We need to find the optimal number of LDA clusters and want to compare the results of our LDA and NMF models. As NMF is not a probabilistic model, it may be difficult to use perplexity to evaluate both models. Therefore, we chose the coherence score

## Milestone II - braedon-patrikt-puzeng

for the evaluation. However, as lyrics are short and informal documents, the traditional coherence score based on the concurrence of frequent words between two documents may not be appropriate. In fact, the traditional coherence scores are so low (almost -60), and we need to find other coherence metrics. Therefore, we use coherence scores based on word embeddings such as Glove (TC-W2V) instead of the traditional one so that the semantic similarities are taken into account when calculating the similarity of two documents, and reasonable results are achieved. The formula to calculate TC-W2V is below.

$$TC - W2V = \frac{1}{C_n^2} \sum_{j=2}^N \sum_{i=1}^{j-1} \text{Similarity}(wv_j, wv_i)$$

We found the optimal number of clusters for the LDA model is 8. After choosing the optimal number of topics, we analyzed the sensitivity of the coherence scores with different combinations of prior document-topic and topic-word distributions. As the default value of these parameters in the scikit-learn is  $1/n\_components$ , which is 0.125 in our case, we selected the range of 0.05 to 0.2 for our sensitivity analysis.

Our model achieved a TC-W2V score of 0.611 with the default setting, where both prior parameters are 0.125, and we can see from the table below that the score is robust if we change the hyperparameters for prior topic-word and doc-topic distributions.

	topic_word_prior						
		0.050	0.100	0.150	0.200	0.250	0.300
doc_topic_prior	0.050	0.614	0.613	0.614	0.614	0.614	0.614
	0.080	0.619	0.618	0.618	0.614	0.619	0.613
	0.110	0.613	0.613	0.613	0.592	0.588	0.588
	0.140	0.608	0.609	0.608	0.609	0.604	0.602
	0.170	0.605	0.605	0.597	0.596	0.596	0.598
	0.200	0.592	0.592	0.591	0.590	0.592	0.592

Table 6 Sensitivity analysis

## C. Evaluation

Though perplexity is commonly used in the NLP field to evaluate probabilistic topic models, we found this score increases with the number of topics, forming an abnormal perplexity-number\_clusters curve. Jiang et al.[7] faced the same problems when they applied LDA models on Twitter corpus, and Chang et al. [8] have concluded that predictive likelihood and human judgment of the quality of learned topics are often not correlated. Perhaps the informal nature and short lengths of lyrics and Twitter corpus are not best for perplexity calculation, so we use manual evaluation, which is the main method used by Jiang et al., and coherence score to evaluate our model.

TC-W2V calculates each cluster's average cosine similarity of the high-weighted topic words to evaluate the coherence. Our optimal model results in a TC-W2V score of 0.61, meaning the top topic words in each cluster are highly correlated.

Another evaluation technique we took was exploring the visualization of 8-topic clusters. We checked the size and overlap of our topic clusters. The two closest clusters were cluster 4 and cluster 6 and this checks out because the main topic for cluster 4 was talking about life and death, and cluster 6 was about god. Both of them may focus on the meaning of life. The most dissimilar topics were cluster 3 and cluster 6. This is because cluster 3 uses harmful lyrics typically found in rap songs, and cluster 6 is about god and Jesus. The largest cluster from our LDA model is cluster 7, which is about love and time, and our smallest cluster is topic 0, which is about love and girls. From the UMAP visualization, the clusters are far from each other, and there's no overlap between them, which further verifies the conclusion we got from the TC-W2V score that the model resulted in distinctive clusters.

## Milestone II - braedon-patrikt-puzeng

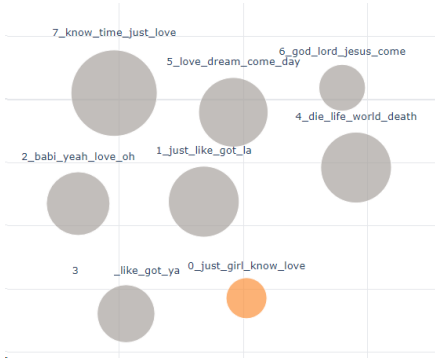


Figure 9 UMAP of LDA results



Figure 10 Word cloud of cluster 4



Figure 11 Word cloud of cluster 6

### 3. TOPIC DETECTION THROUGH NON NEGATIVE MATRIX FACTORIZATION

#### A. Structure

The NMF model finds the latent structure by decomposing the corpus as a non-negative matrix into two factors, which excels at discovering topics within large text-filled datasets. By utilizing this method from the scikit-learn, we discovered the significant topics from the lyrical song data set. This is a unique method compared to others used because it is a nonprobabilistic model.

#### B. Experiments

We use the same topic range to find the optimal number of topics by comparing the TC-W2V scores, and the result is slightly different from what we find with LDA.

#### C. Evaluation

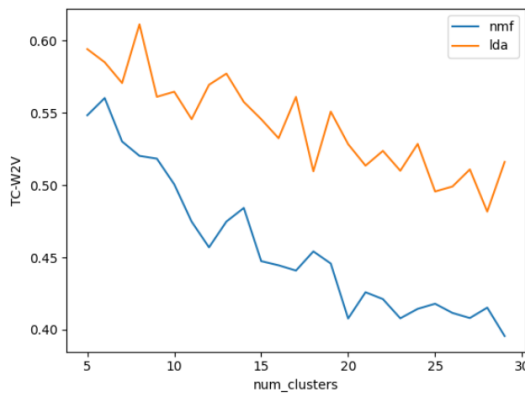


Figure 12 Scores and number of clusters

The TC-W2V score of our model is 0.56, which is meaningful but not as significant as the score achieved by the LDA model. It's interesting to find that LDA scores are higher than NMF from the TC-W2V and number of topics plot. Perhaps one important difference between lyrics and normal corpora, such as news and Wiki pages, is that there are multiple or even contradictory emotions in the same song. For example, some songs about the memory of a former lover sound agreeable and melancholy. Therefore, the assumption of the LDA model that each corpus has many topics may be more appropriate in this situation.

An important evaluation technique we took forth was manually looking at the visualization of the 6 topics. The two closest topics based on distance were topic 0 and topic 1. This makes sense because the main topic for 0 was talking about love, and topic 1 was about like and getting. The most dissimilar topics were topics 0 and 3. This is because topic 0 focuses on love, while topic 3 is some offensive rap lyrics. Interestingly, our NMF topic results show that the largest topic is different from the LDA model. Based on diameter, the largest topic is topic 4, the harmful rap lyrics. The clusters in UMAP visualization are distinctive as there's no overlap between them.



Figure 13 UMAP of NMF results



Figure 14 Word cloud of cluster 0



Figure 15 Word cloud of cluster 1

## VII. DISCUSSION

### 1. SUPERVISED LEARNING

In our attempt to classify a song's genre and emotion, we used several different approaches.

- 1) A classifier trained to classify a song's genre based on a dataset containing audio features will outperform random guessing, but its peak performance is not phenomenal. Using a more complex model, such as a variant of a neural network, is a more effective approach to the genre classification problem.
- 2) The most challenging thing in our supervised learning task is to find a good representation of music data. We have tried different formats, such as the average MIDI intensity matrix of all instruments, of piano, of the most intense instrument, MFCCs, and Mel Spectrum. The result is that unexpectedly, a simple data structure like our derived piano intensity matrix can result in good accuracy.
- 3) We found it interesting that our CNN and RNN genre classification models have almost the same accuracy, which may imply that the bottleneck to improve the performance next is related to feature engineering.
- 4) We also tried different methods to detect the emotion or topic of music to generate labels for later classification tasks. Average word embeddings from pre-trained models, from models trained by ourselves, and LLM results were all experimented with, while the former results are too diverse for classification. We also tried to regress the average lyric word embeddings from the pre-trained model with different music representations, but the results are too general, perhaps because of our limited dataset and large result space.

Further study can focus on these aspects based on our model:

- 1) Use more representative data if there is access to sufficient memory. Our results were achieved using only the average music notes from the piano instruments, but perhaps a better accuracy can be achieved if different categories of instruments are represented in different layers. This is especially helpful in identifying Rock, Electronic, and other more instrumentally focused genres. However, 80 GB of memory may be needed for the same dataset.
- 2) A better LLM can be used to generate the labels for emotion classification tasks. As mentioned above, the Flan-T5-Base may not be able to generate the ideal label even if we add additional constraints in the prompt (providing an emotion list or limiting the length of the result), but it seems not to be a problem for Chat-GPT.
- 3) We only used traditional CNN and RNN structures in this project. As the transformer has dominated the field of NLP and shown great success in the CV area, perhaps a more advanced model based on the transformer can achieve better accuracy and combine the analysis of music notes, lyrics, and MFCCs together.

### 2. UNSUPERVISED LEARNING

Our attempt to decompose lyrics into topics is both interesting and challenging.

- 1) We are surprised to find that the optimal number of clusters is quite similar in all three methods. Considering the number of emotions generated by the Flan-T5 model in our supervised learning section is also not too far away, perhaps these results reflect some nature of lyric clustering.
- 2) The most challenging part of our unsupervised learning study is to find a good evaluation metric for the informal corpus. It's difficult to apply perplexity to non-probabilistic methods such as NMF, and traditional coherence metrics may not be appropriate for short corpus. Therefore, after reading some materials, we chose TC-W2V for evaluation.

Perhaps we would try these aspects if given more time.

- 1) As transformers have dominated the field of NLP, perhaps pre-trained transformers such as BERT, Flan-T5, and GPT-4 can be used to detect the topics with a small amount of training. We would be happy to use the state-of-the-art transformer model for our lyric analysis if more time and resources are available.
- 2) We only focus on the English songs in our project, ignoring the diversity of languages and cultures. If we have more time, we would like to extract topics from lyrics of different languages and translate them into English with some classical machine learning methods. One optional translation method[9] that DeepLearning.AI used in their NLP courses is to train a linear conversion matrix or a neural network to transform pre-trained word embeddings of other languages to English pre-trained word embeddings, and we can use this model to find corresponding English topic words or emotion words from words in other languages.

## VIII. ETHICAL CONSIDERATIONS

### 1. SUPERVISED LEARNING

Our MIDI dataset contains most genres but not in a large enough sample size to incorporate genres that are unique to specific cultures and languages. As most of our team members are Americans, we are analyzing only English lyrical songs. We think the root cause of this lack of representation of other languages is that when we are searching for accessible datasets that meet the

## Milestone II - braedon-patrikt-puzeng

project requirements, it's done through a prominent United States-based platform. Therefore, most of the data posted will be from Americans, and Americans rarely speak other languages. So, in the end, the data available to them is in English, and then it's posted in English predominantly, and then we analyze it. This loop of focusing solely on English-based data could be solved by considering other similar data set websites, such as CodaLab, Zind, IDA, Signate, Tianchi, and ExploreSA. In principle, each one of these companies is almost identical to our data sources but is prominent in Europe, Africa, Russia, China, Japan, and Australia. Therefore solving the ethical issue of underrepresentation of other cultures in this report.

## 2. UNSUPERVISED LEARNING

The potential ethical issues arising from our unsupervised clustering stems from harmful or offensive language. By incorporating music that uses words that have historically been used negatively towards certain races, presenting the raw results to the general public is unethical. Before presenting our unsupervised findings, we would need to carefully alter some topic vocabulary to allow for an inclusive and supportive report for people with all backgrounds. A way to address this would be to modify our English stopword function to eliminate the lyrical words from the dataset or even swap the harmful words for more socially acceptable phrases.

## IX. STATEMENT OF WORK

Task		Pu Zeng	Braedon Shick	Patrick Thornton
Preprocessing and EDA	Data Cleaning	√	√	√
	MIDI/MP3 transformation	√		
	Data Exploration	√	√	√
Unsupervised Learning	PCA		√	
	LDA		√	
	NMF		√	
	Model Evaluation		√	
Supervised Learning	Random Forest			√
	Gradient Boost			√
	SVM			√
	Others (Naive Bayes, SGD, Logistic Regression)			√
	LSTM (Genre Classification)	√		
	CNN (Genre Classification)	√		
	CNN (Emotion Regression)	√		
	CNN (Emotion Classification)	√		
	Model Evaluation	√		√
Report Writing	Report	√	√	√

Table 7 statements of work



## X. REFERENCES

- [1] Ferreira, Lucas, Levi Lelis, and Jim Whitehead. "Computer-generated music for tabletop role-playing games." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 16. No. 1. 2020.
- [2] Raffel, Colin. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. 2016." (2016).
- [3] Bertin-Mahieux, Thierry, et al. "The million song dataset." (2011): 591-596.
- [4] Soleymani, Mohammad, et al. "1000 songs for emotional analysis of music." *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*. 2013.
- [5] Ferreira, Lucas N., and Jim Whitehead. "Learning to generate music with sentiment." *arXiv preprint arXiv:2103.06125* (2021).
- [6] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
- [7] Bian, Jiang, et al. "Mining Twitter to assess the public perception of the “Internet of Things”." *PloS one* 11.7 (2016): e0158450.
- [8] Chang, Jonathan, et al. "Reading tea leaves: How humans interpret topic models." *Advances in neural information processing systems* 22 (2009).
- [9] Natural Language Processing Specialization, [www.deeplearning.ai/courses/natural-language-processing-specialization/](https://www.deeplearning.ai/courses/natural-language-processing-specialization/). Accessed 23 Oct. 2023.

## XI. APPENDIX

### APPENDIX I SUBMITTED CODE FILE

```

└─Data_Preprocessing    # Our data preprocess scripts, which is called in the Jupyter Notebook
|   Emotion_Label.py
|   Emotion_MIDI2.py
|   MP3_Emotion.py
|   Piano_Genre_Classification.py
|
└─FINAL_CODE_For_Our_Toy_Example # Our Final code developed with Google Colab
|   Supervised 01 - Spotify.ipynb
|   Supervised 02 - CNN_Genre_Classification.ipynb
|   Supervised 03 - RNN_Genre_Classification.ipynb
|   Supervised 04 - CNN_Emotion_Classification_With_Flank-T5_Labels.ipynb
|   Supervised 05 - CNN_Emotion_Regression_MIDI.ipynb
|   Supervised 06 - CNN_Emotion_Regression_MP3.ipynb
|   Unsupervised 1 - TSNE on Glove.ipynb
|   Unsupervised 2 - LDA and NMF.ipynb
|
└─Report Resources      # Notebooks ran on the entire dataset, which is only used to show the report results
|   Supervised 01 - Spotify.ipynb
|   Supervised 02 - CNN_Genre_Classification.ipynb
|   Supervised 03 - RNN_Genre_Classification.ipynb
|   Supervised 04 - CNN_Emotion_Classification_With_Flank-T5_Labels.ipynb
|   Supervised 05 - CNN_Emotion_Regression_MIDI.ipynb
|   Supervised 06 - CNN_Emotion_Regression_MP3.ipynb
|   Unsupervised 1 - TSNE on Glove.ipynb
|   Unsupervised 2 - LDA and NMF.ipynb
|
└─Toy_Dataset

```

### APPENDIX II COLAB LINK

**We strongly recommend running our code in Google Colab, as some extra libraries need to be installed, and some of them need exe installation (Appendix VI).**

The Code is tested under Google Colab with both CPU and T4 environments, while some of which may not be able to run completely due to the limit of the memory. We strongly recommend using Colab to run our code. Please use your UM Email to open the Colab links below.

#### Supervised learning

##### 1. Spotify Classification

Only CPU is needed.

[https://colab.research.google.com/drive/1ZXcR0mMVhKqa9xc0EJSdf\\_8ezs6dncDN](https://colab.research.google.com/drive/1ZXcR0mMVhKqa9xc0EJSdf_8ezs6dncDN)

Milestone II - braedon-patrikt-puzeng

## 2. CNN Genre Classification

We recommend using T4 for this code.

<https://colab.research.google.com/drive/1mM1gr5zNGnKEmYkOrF5Rdiw6beGFqpMM>

## 3. RNN Genre Classification

We trined this model with Colab V100 in HIGH-RAM mode. Both CPU or T4 will take a super long time to train the model and perhaps run out of memory if not in HIGH-RAM mode. If your are not a Colab Pro, please stop at the first validation stage, and uncomment the cell below to use the first model for evaluation.

<https://colab.research.google.com/drive/1PHJQzMdMgTzsmX-byRvSU8atASMQyLDT>

## 4. CNN Emotion Classification

We recommend using T4 for this notebook, as transformers are needed to generate the labels.

Due to the size of this model, the evaluation code of the ensembled model may only work with Colab HIGH-RAM mode. If you are not a Colab Pro, Please use only one model for evaluation.

<https://colab.research.google.com/drive/11hK8EbE94ZXcOvYemIleYNVEHwC0VzWQ>

## 5. CNN Regression with MIDI files

Both CPU and T4 works well for this notebook, while T4 is significantly faster.

<https://colab.research.google.com/drive/1kyOV-ov1QQsJ2df-u6rK1447iZLDzq5x>

## 6. CNN Regression with MP3 files

Both CPU and T4 works well for this notebook, while T4 is significantly faster.

<https://colab.research.google.com/drive/1Up8aXdtNe5ho945bja13-Nx2v6u7ut6m>

## Unsupervised learning

GPU is not used in this notebook, so there's no need to use T4.

To see the UMAP plot of the clusters, please click on the link generated by the NMF and LDA code.

After installing the visualization library, Colab need to be restarted to work properly. We've added some code to restart the notebook after the library install. However, you need to run the "import library" cell again after the restart.

### 1. TSNE on Average Glove Embeddings

<https://colab.research.google.com/drive/1BCZr--eTRAna5L54C4b3jESrbjMdoknC>

### 2. LDA and NMF on Lyrics

<https://colab.research.google.com/drive/1YbBROY315WVXjzLUXgZMTn1PehCgKCId>

## APPENDIX III FEATURE LIST OF THE SPOTIFY DATASET

Feature	Explanation	Type
track_popularity	Song Popularity (0-100) where higher is better	float
danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.	float

## Milestone II - braedon-patrikt-puzeng

energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.	float
key	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C#/D ♭, 2 = D, and so on. If no key was detected, the value is -1.	float
loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.	float
mode	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.	float
speechiness	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.	float
acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.	float
instrumentalness	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.	float
liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.	float
valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).	float
tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.	float
duration_ms	Duration of song in milliseconds	float

Appendix Table 1 Spotify Features

APPENDIX IV CNN MODEL STRUCTURE

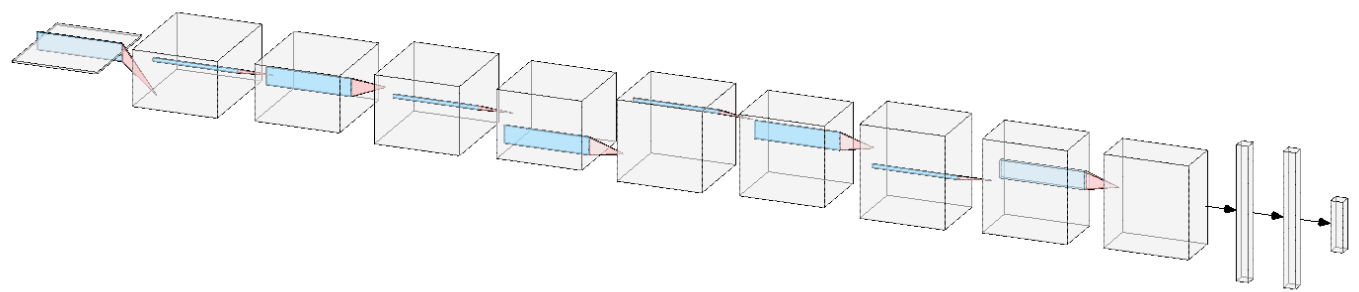
1. CNN for Genre Classification

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 24, 503, 128]	264
ReLU-2	[-1, 24, 503, 128]	0
MaxPool2d-3	[-1, 24, 251, 128]	0
Conv2d-4	[-1, 48, 242, 128]	11,568
ReLU-5	[-1, 48, 242, 128]	0
MaxPool2d-6	[-1, 48, 121, 128]	0
Conv2d-7	[-1, 96, 112, 128]	46,176
ReLU-8	[-1, 96, 112, 128]	0
MaxPool2d-9	[-1, 96, 56, 128]	0
Dropout-10	[-1, 96, 56, 128]	0
Conv2d-11	[-1, 192, 47, 128]	184,512
ReLU-12	[-1, 192, 47, 128]	0
MaxPool2d-13	[-1, 192, 23, 128]	0
Conv2d-14	[-1, 192, 14, 127]	737,472
ReLU-15	[-1, 192, 14, 127]	0
MaxPool2d-16	[-1, 192, 7, 127]	0
Linear-17	[-1, 300]	51,206,700
ReLU-18	[-1, 300]	0
Dropout-19	[-1, 300]	0
Linear-20	[-1, 17]	5,117

Total params: 52,191,809  
Trainable params: 52,191,809  
Non-trainable params: 0

Input size (MB): 0.25  
Forward/backward pass size (MB): 117.78  
Params size (MB): 199.10  
Estimated Total Size (MB): 317.12

Appendix Table 2 CNN Genre Classification Model Size



Appendix Figure 1 CNN Genre Classification Model Structure

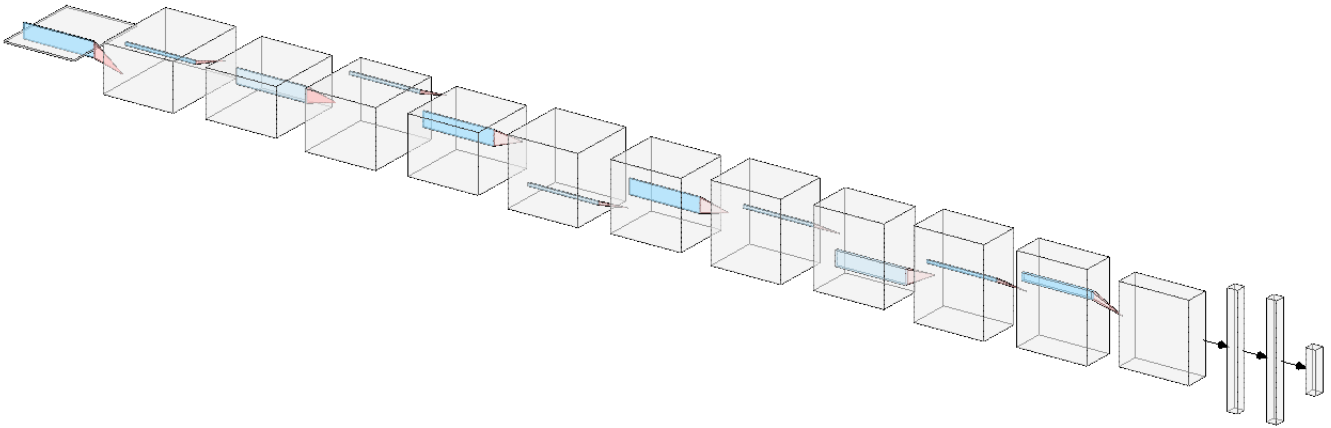
## 2. CNN for Emotion Classification

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 24, 503, 128]	264
ReLU-2	[-1, 24, 503, 128]	0
MaxPool2d-3	[-1, 24, 251, 128]	0
Conv2d-4	[-1, 48, 242, 128]	11,568
ReLU-5	[-1, 48, 242, 128]	0
MaxPool2d-6	[-1, 48, 121, 128]	0
Conv2d-7	[-1, 96, 112, 128]	46,176
ReLU-8	[-1, 96, 112, 128]	0
MaxPool2d-9	[-1, 96, 56, 128]	0
Dropout-10	[-1, 96, 56, 128]	0
Conv2d-11	[-1, 192, 47, 128]	184,512
ReLU-12	[-1, 192, 47, 128]	0
MaxPool2d-13	[-1, 192, 23, 128]	0
Conv2d-14	[-1, 384, 19, 127]	737,664
ReLU-15	[-1, 384, 19, 127]	0
MaxPool2d-16	[-1, 384, 9, 127]	0
Conv2d-17	[-1, 192, 5, 126]	737,472
ReLU-18	[-1, 192, 5, 126]	0
MaxPool2d-19	[-1, 192, 2, 126]	0
Linear-20	[-1, 4096]	198,184,960
ReLU-21	[-1, 4096]	0
Dropout-22	[-1, 4096]	0
Linear-23	[-1, 11]	45,067

Total params: 199,947,683  
 Trainable params: 199,947,683  
 Non-trainable params: 0

Input size (MB): 0.25  
 Forward/backward pass size (MB): 131.05  
 Params size (MB): 762.74  
 Estimated Total Size (MB): 894.04

Appendix Table 3 CNN Emotion Classification Model Size



Appendix Figure 2 CNN Emotion Classification Model Structure

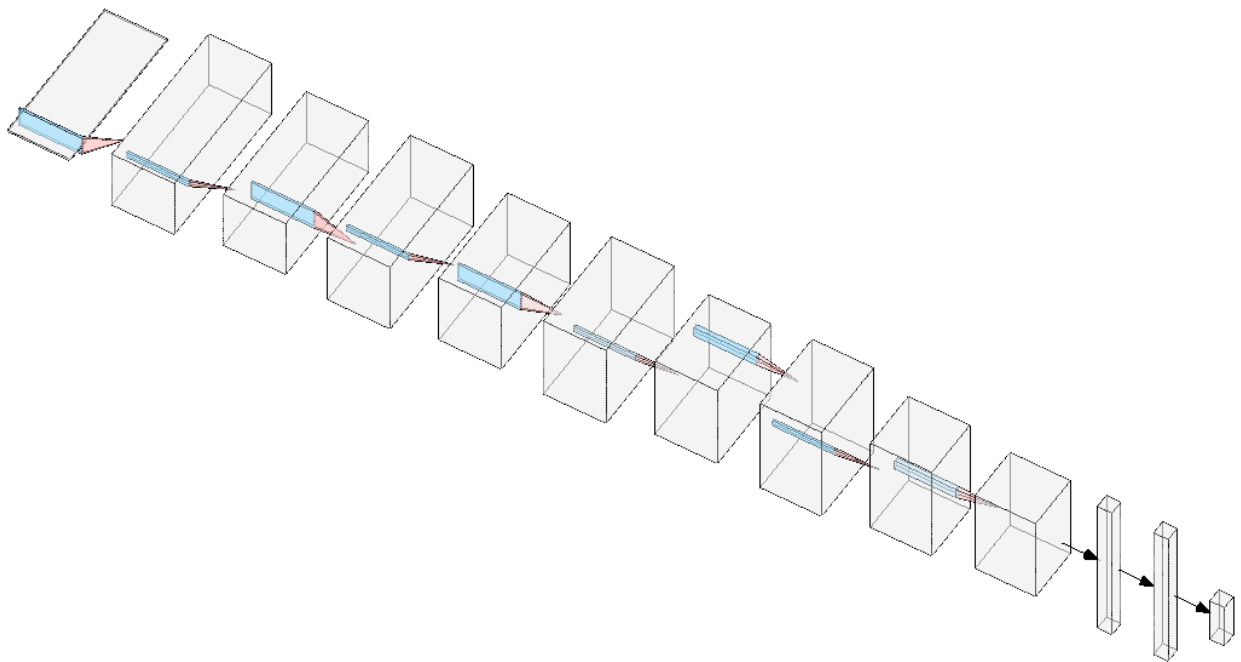
### 3. CNN for Regression

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 24, 1913, 20]	264
ReLU-2	[-1, 24, 1913, 20]	0
MaxPool2d-3	[-1, 24, 637, 20]	0
Conv2d-4	[-1, 48, 628, 20]	11,568
ReLU-5	[-1, 48, 628, 20]	0
MaxPool2d-6	[-1, 48, 209, 20]	0
Conv2d-7	[-1, 96, 200, 20]	46,176
ReLU-8	[-1, 96, 200, 20]	0
MaxPool2d-9	[-1, 96, 66, 20]	0
Dropout-10	[-1, 96, 66, 20]	0
Conv2d-11	[-1, 192, 57, 19]	368,832
ReLU-12	[-1, 192, 57, 19]	0
MaxPool2d-13	[-1, 192, 19, 19]	0
Conv2d-14	[-1, 96, 15, 18]	184,416
ReLU-15	[-1, 96, 15, 18]	0
MaxPool2d-16	[-1, 96, 5, 18]	0
Conv2d-17	[-1, 96, 1, 17]	92,256
ReLU-18	[-1, 96, 1, 17]	0
Linear-19	[-1, 64]	104,512
ReLU-20	[-1, 64]	0
Dropout-21	[-1, 64]	0
Linear-22	[-1, 2]	130

Total params: 808,154  
 Trainable params: 808,154  
 Non-trainable params: 0

Input size (MB): 0.15  
 Forward/backward pass size (MB): 39.06  
 Params size (MB): 3.08  
 Estimated Total Size (MB): 42.29

Appendix Table 4 CNN Regression Model Size



Appendix Figure 3 CNN Regression Model Structure

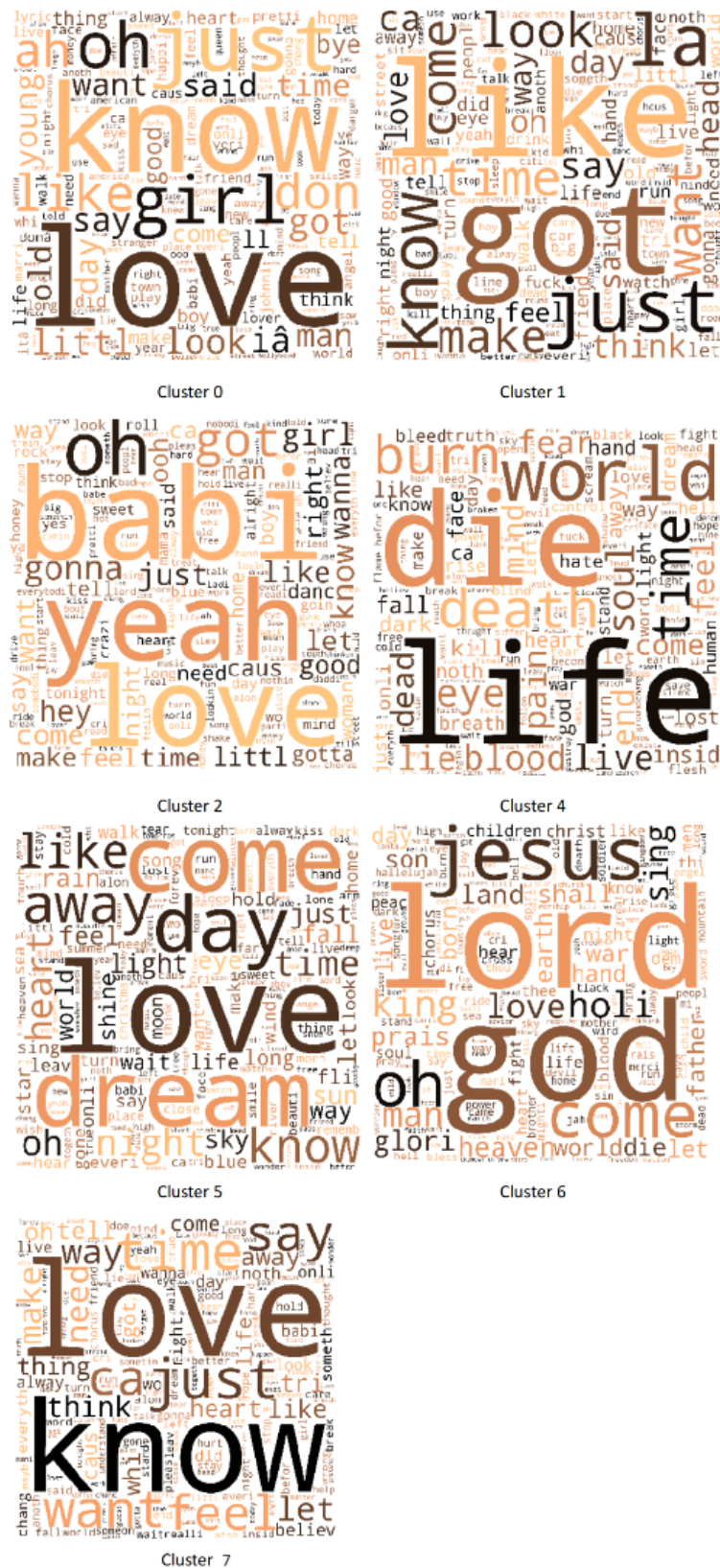


## APPENDIX V UNSUPERVISED LEARNING RESULTS

### 1. Word Clouds of the NMF Model (Swear word cluster is deleted)



## 2. Word Clouds of the LDA Model (Swear word cluster is deleted)






## APPENDIX VI HOW TO INSTALL FLUIDSYNTH IN WINDOWS

In our CNN regression experiments, we need to transform a MIDI file into a WAV file so that the MFCCs can be extracted. However, installing FluidSynth is a little tricky in Windows. Here is a brief introduction of how to install FluidSynth. For other systems, please refer to the website <https://github.com/FluidSynth/fluidsynth/wiki/Download>.

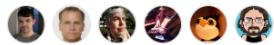
1. To install FluidSynth, please download the newest binary file for Windows 10 from this website:

<https://github.com/FluidSynth/fluidsynth/releases/tag/v2.3.4>

 derselbst released this last month  v2.3.4  5ecd45






- Fix a build failure when specifying `CHAKE_INSTALL_LIBDIR` as an absolute path (#1261, thanks to @OPNA2608)
- Fix some MIDI files never finish playing (#1257, thanks to @joanbm)
- Implement IPv6 to IPv4 fallback (#1208, thanks to @ivan-zaera)
- Fix a build failure when using CMake's Xcode generator (#1266, thanks to @bradhowes)
- Fix pipewire's Jack implementation not found by CMake (#1268, thanks to @pedrolcl)
- Fix a regression causing the MIDI Player to terminate prematurely (#1272, thanks to @albedozero)

### Contributors



joanbm, bradhowes, and 4 other contributors

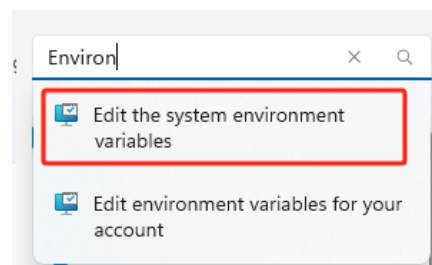
### ▼ Assets 5

 fluidsynth-2.3.4-android24.zip	35.6 MB	last month
 fluidsynth-2.3.4-win10-x64.zip	5.77 MB	last month
 fluidsynth-2.3.4-winXP-x86.zip	3.44 MB	last month
 Source code (zip)		last month
 Source code (tar.gz)		last month

 4  2 6 people reacted

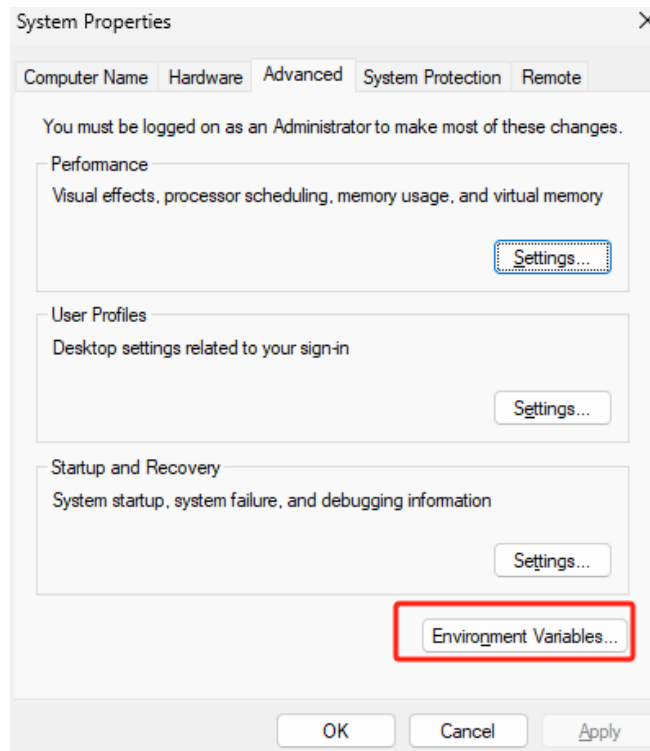
Appendix Figure 6 Download FluidSynth

- Unzip the file.
- Open the “Edit the system environment variable” in settings.



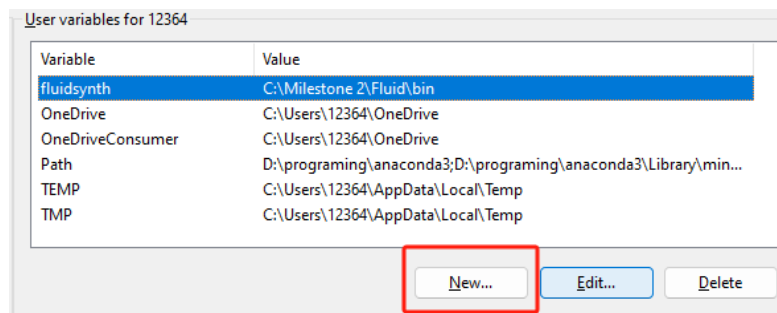
Appendix Figure 7 Open the environment variable setting

- Click on settings.



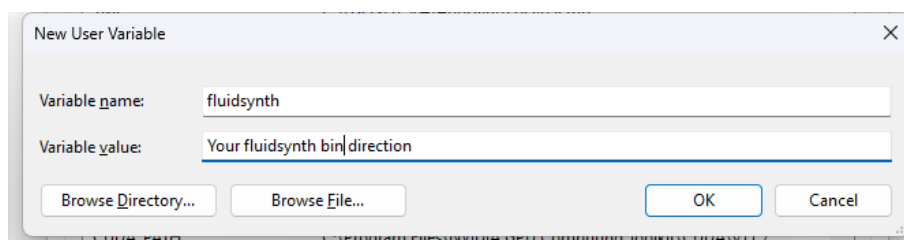
Appendix Figure 8 Click on Environment Variables...

5. Click on the New



Appendix Figure 9 Click on New

6. Set the variable to your fluidsynth bin directory.



Appendix Figure 10 Add Variables

7. Download the SoundFont files from the website <https://github.com/FluidSynth/fluidsynth/wiki/SoundFont>

8. Copy the SoundFont file into the fluidsynth bin directory and User/{Username}/.fluidsynth directory.