Piano Music Sentiment Classification
CIS 522 Final Project

Bachpropagation

April 28, 2022

Team Members:

- Jason Shu; jasonshu; Email: `jasonshu@seas.upenn.edu`
- Joseph Poirier; jcpoir; Email: `jcpoir@seas.upenn.edu`
- Paul Scott; pscott4; Email: `pscott4@seas.upenn.edu`

Abstract

While state-of-the-art deep learning techniques have been used to classify sequences of text data by sentiment, similar techniques may be used for music, especially as music streaming platforms become increasingly ubiquitous and reliant on artificial intelligence to recommend music to users and understand their preferences. Using classical piano MIDI recordings from the MAESTRO dataset and emotion labels (danceability, energy, and valence) queried using Spotify's Web API, we trained logistic regression, plain RNN, LSTM, and transformer models to classify human performances of classical music by their emotional content. The first 100 notes of each piece are considered, and for each note, models predict based on the pitch, duration, and step (time since the last note started), as well as a length-12 vector encoding the harmonic context or chord associated with the note. Bidirectional LSTMs were found to greatly outperform the other baseline models in all three classification tasks; removing the harmony attributes and just using the raw MIDI data decreased accuracy for energy and valence while increasing accuracy for danceability. While the transformer model did not perform as well as the other models, it clearly requires more data to be trained properly and the general approach shows promise as shown by projects like Microsoft's MusicBERT as well as MIDI-BERT. The methods described in this paper can be applied to arbitrary musical recordings in a MIDI format, as well as any other formats after converting to MIDI.

1 Introduction

Music is a powerful medium for expressing emotion — often called a universal language, it can indeed be a way for humans to wordlessly communicate by drawing upon shared cultural and evolutionary history. Now, as our exposure to music becomes increasingly digital, automated methods to process, to understand, and to generate music have become of interest in the AI community, mirroring a similar but more widespread trend in natural language domains. For music streaming giants like Spotify, YouTube, and Apple Music, it is particularly important to have a scalable way to understand the emotional content of a piece of music in order to infer users' listening habits and recommend new music to them in real time.

We wanted to analyze expressive human performances of potentially complex non-verbal music — while there exist NLP algorithms to gauge sentiment of song lyrics [NS18], we focused on music more defined by its non-verbal aspects like melody, harmony, and rhythm (which can still be approached with language model techniques). This eventually led us to a classical piano performance dataset, MAESTRO [Haw+19]. The MAESTRO dataset contains audio and MIDI files of classical piano pieces performed by professional pianists. Comparable papers focused on music sentiment analysis [QCZ22] [Cho+21] often use popular piano music datasets like EMOPIA, which tends to be less harmonically and texturally complex than the music in the MAESTRO dataset. To further narrow our focus on the music itself, we chose to work with the MIDI format, which is much more space- and resource-efficient than audio (even if it does not perfectly capture the original sound of the performance). For sentiment labels, we leveraged Spotify's Web API through the Python library Spotipy [Lam+]. More details regarding the data can be found in Section 3.

Below, we run several classes of models — logistic regression, recurrent neural net (RNN), transformer, and long short-term memory (LSTM) — which predict the sentiment of a piece in three different senses: danceability (how easy it is to dance to the music), energy (how energetic the music is), and valence (how happy or sad the music sounds). The models are trained and tested on the first 100 notes of pieces, using only the pitch, duration, step (time difference between consecutive notes), and a custom-extracted harmony feature for each note. It was found that LSTMs vastly outperform logistic regression, transformer, and RNN models in all three tasks, while logistic regression slightly outperforms RNNs. Transformers underperform in all areas, likely due to overfitting. We also removed the harmony feature to see how the LSTM fares on just the raw MIDI as downloaded from the website, and it was found that it performs much worse for predicting valence and energy, but slightly better for danceability.

2 Related Work

Recently, many groups have sought to develop algorithms that can reliably predict sentiment from audio or structured music data. In addition to melody recognition, sentiment analysis can be useful in interpreting and even recommending music.

In "Multi-Modal Music Emotion Recognition: A New Dataset, Methodology, and Comparative Analysis", researchers apply a variety of shallow architectures to the MIREX Mood Classification task, which entails performing sentiment analysis on a set of lyric and midi files. The group found that incorporating melodic audio features into their sentiment prediction algorithms helped across all algorithms. Of the shallow models implemented in the study, a support vector machine model performed the best on this task — scoring 64% on the sentiment analysis task, compared to 38.3% scored by a naïve Bayes classifier [**MER**].

BERT, which has helped in natural language processing with tasks such as language translation and text sentiment analysis, has recently found multiple applications in music sentiment analysis.

In "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding", researchers pretrain a 12-layer transformer model on a mixture of "polyphonic piano MIDI files," ranging from Japanese anime music to Korean and Western pop and Western classical. The group then fine-tuned their transformer model for four tasks: melody extraction, velocity prediction, composer classification, and emotion classification. The group found that their transformer could outperform an RNN baseline for all four tasks [Cho+21].

A similar study out of Microsoft, "MusicBERT: Symbolic music Understanding with Large-Scale Pre-Training", implemented a transformer encoder model on a large music dataset. The primary dataset in this study contained symbolic music from over 1.5 million songs with a rich array of features, including bar position, tempo, instrument, velocity, and pitch. Two models were implemented in the study; a "small" model, with four attention layers with eight heads each, and a "base" model, which used 12 layers and 12 heads. With pretraining, MusicBERT scored 96.7% on a melody prediction task [Zen+21].

3 Dataset and Features

The MIDI and Audio Edited for Sychronous TRacks and Organization (MAESTRO) dataset is a collection of 200 hours (1184 performances) of virtuoso piano music. The dataset contains .wav and midi files for each of the performances, along with metadata including the canonical title and composer of the song, the year of the performance, the midi and audio file names and the duration of the song. For this study, MIDI files were used for all prediction tasks.

As the files in the MAESTRO dataset are unlabeled, canonical titles and composers were used to query sentiment scores from from the Spotify API. This study is limited to predicting energy, danceability, and valence, so these attributes only were queried from Spotify.

MIDI files are structured so that a note is represented by its order, pitch, step, duration and velocity. Velocity in this case refers to the force with which a note is played, not the tempo of the song or timing of the note. Pitch, originally recorded as an integer on [0, 128], was standardized to float values on [0, 1].

Sentiment attributes, queried from Spotify as floats on [0, 1], were converted to ordinals from one to three based on their values. In the case of danceability, for example, a value of one represents a not-very danceable song (0 - 0.33), a value of two represents a somewhat danceable song (0.33 - 0.67), and a value of three represents a very danceable song (0.67 - 1). All three attributes were converted in this manner and stored in a set of of three n x 3 label arrays.

For training, the pitch, step, and duration of the first 100 notes of each song were standardized and distributed among train and test datasets. Unlabeled songs or songs with fewer than 100 notes were omitted. Eight-hundred and fifty songs were ultimately allocated to the training dataset, with the remaining 213 held out for testing. Hyperparameter optimization was conducted using cross-validation over the training set, where applicable. For training without harmony, this data was stored in a set of three n x 100 x 3 arrays.

Harmony data was added as a one-hot vector (more specifically, a multi-label binarized vector) of length 12, where each slot represents a pitch in the standard Western octave (e.g. C, C-sharp, D, etc.). Octave position was disregarded. Harmonies associated with each note in the MIDI were defined as the group of up to three notes that have the greatest overlap with the primary note (encoded in the "pitch" column); each such harmony note is a 1 or 0, depending on whether it is included or excluded. Where notes shared overlap with only two or fewer other notes, only those notes were encoded as 1 in the harmony vector. The resulting harmony training array was n x 100 x 15, with 3 spots from the original dataset and 12 allocated to harmony data.

4 Methodology

For each of the following model types, we train three separate models, one for each sentiment class.

4.1 Logistic Regression

For our baseline model we used logistic regression. Because this model expects data to be two-dimensional and our data is three-dimensional, we flatten our data from the shape n x 100 x 15 to n x 1500 (where n is the batch size). We performed 10-fold cross-validation to find the optimal values for the model's hyperparameters, which for this model is only the penalty amount, C. The following values of C were testing during cross-validation:

$$C = [0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 5.0, 10.0]$$

After running cross-validation, we found that the optimal C value for all three sentiment classes was $C = 0.05$.

4.2 RNN

The second model that we tested was an RNN model. This model is better suited for sequential data, unlike logistic regression. The model consists of multiple RNN layers with an output dense layer with 3 units and with softmax activation. We trained the RNN using categorical cross-entropy loss and using the Adam optimizer with a learning rate of 0.001. We performed 4-fold cross-validation over several hyperparameters, including the number of RNN layers, the number of units per RNN layer, and whether to make the RNN layers bidirectional. The following values for each of these hyperparameters were tested during cross-validation:

$$num\_layers = [1, 2, 3]$$

$$num\_units = [128, 256, 512]$$

$$bidirectional = [False, True]$$

After performing cross-validation, we found the following optimal hyperparameters for each sentiment class:

Sentiment Class	**Layers**	**Units**	**Bidirectional**
Danceability	2	256	True
Energy	1	128	True
Valence	1	256	True

Table 4.1: Optimal hyperparameters for RNN model

4.3 Transformer

The third family of models that we tested was a group of transformer architectures. Due to compute constraints, architectures were limited to a maximum of four encoder units with a maximum of eight attention heads each. A full grid-search would also have been unfeasible due to high compute cost, so informal methods were used. Adam optimization with a learning rate of 0.0001 was used for all three models. Stochastic gradient descent (SGD) and Adam optimization with learning rates of 0.0001, 0.001, and 0.01 were tested over ten epochs with the final architectures. Full encoder-decoder models were tested briefly, but encoder-only models were used. In place of context-sensitive word embeddings, feed-forward linear embeddings were used.

Class	**Embed Layers**	**Attn. Layers**	**Heads**	**Output Layers**
Danceability	2	4	8	3
Energy	2	4	8	3
Valence	2	4	8	3

Table 4.2: Optimal hyperparameters for transformer model

The intent with implementing transformer models was (1) to leverage attention to learn generalizations on sequences of notes with some degree of invarance to order and sequence length, and (2) to leverage the high capacity of transformers by fine-tuning a model that is pretrained on MIDI sentiment tasks. The latter goal was not achieved using this methodology, as checkpoint file corruption issues and concerns over the differences between computer and human-generated MIDIs made fine-tuning MIDI-BERT unviable [Cho+21]. As the dataset in this study is too small to effectively train a high-capacity transformer model, transformer-encoder implementation in this case is intended to act as a proof-of-concept.

4.4 LSTM

The final model that we tested was an LSTM model. This model, like RNNs, is also suited for sequential data, but does not suffer as severely from short-term memory loss as RNNs do. The model consists of multiple LSTM layers with an output dense layer with 3 units and with softmax activation. We trained the LSTM using categorical cross-entropy loss and using the Adam optimizer with a learning rate of 0.001. We performed 4-fold cross-validation over several hyperparameters, including the number of LSTM layers, the number of units per LSTM layer, and whether to make the LSTM layers bidirectional. The following values for each of these hyperparameters were tested during cross-validation:

$$num\_layers = [1, 2, 3]$$

$$num\_units = [128, 256, 512]$$

$$bidirectional = [False, True]$$

After performing cross-validation, we found the following optimal hyperparameters for each sentiment class:

Sentiment Class	**Layers**	**Units**	**Bidirectional**
Danceability	3	512	True
Energy	2	128	True
Valence	3	128	True

Table 4.3: Optimal hyperparameters for LSTM model

As an aside, we also ran the LSTM model on the data without the harmony features. In Section 5 we discuss how the LSTM model outperforms the other models, and therefore we use the LSTM model to observe the effects of the harmony on the overall accuracies for each sentiment class. We did this by removing the harmony features and only leaving the pitch, step, and duration features. We did not perform separate hyperparameter tuning in this instance and instead used the same hyperparameters found using the data with harmony.

5 Results

5.1 Loss and Accuracy Plots

The loss plots for all sentiment classes are interesting in that the test losses skyrocket, rather than decrease. However, the training loss does decrease. The training accuracies for all sentiment classes reach about 100%, but the testing accuracies are significantly lower and stop having major increases after early in training.

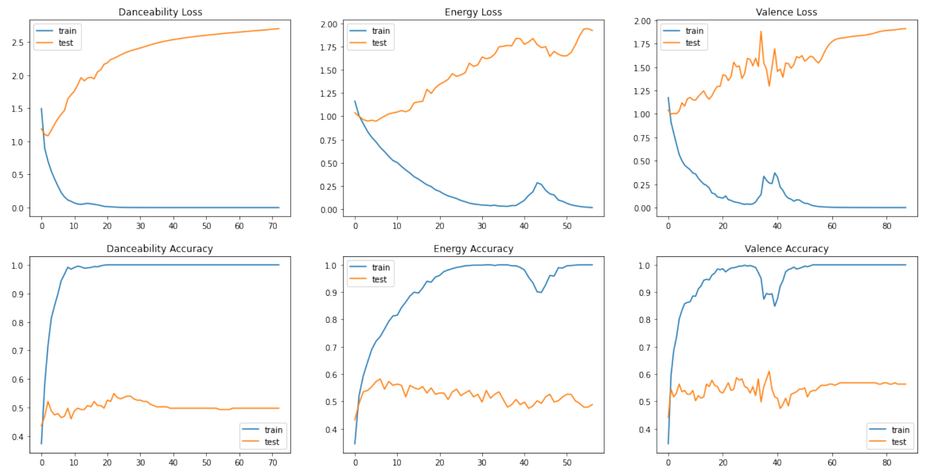

Figure 5.1: Loss and accuracy plots for RNN model

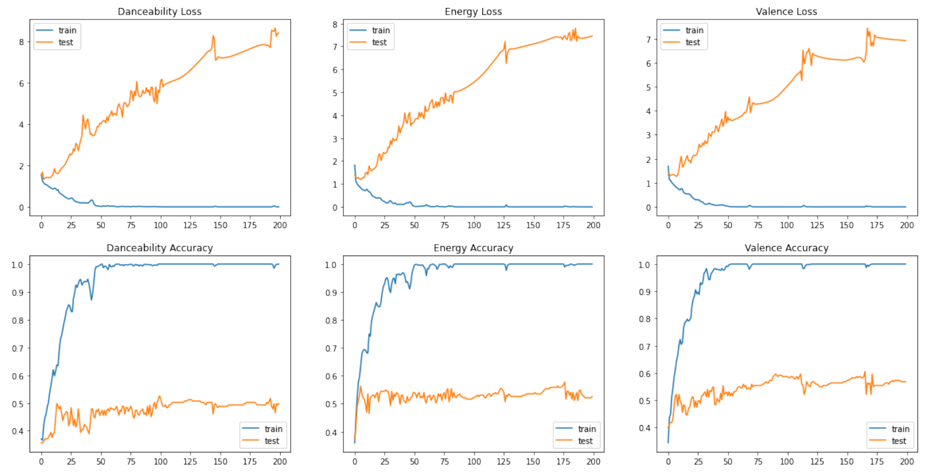

Figure 5.2: Loss and accuracy plots for transformer model

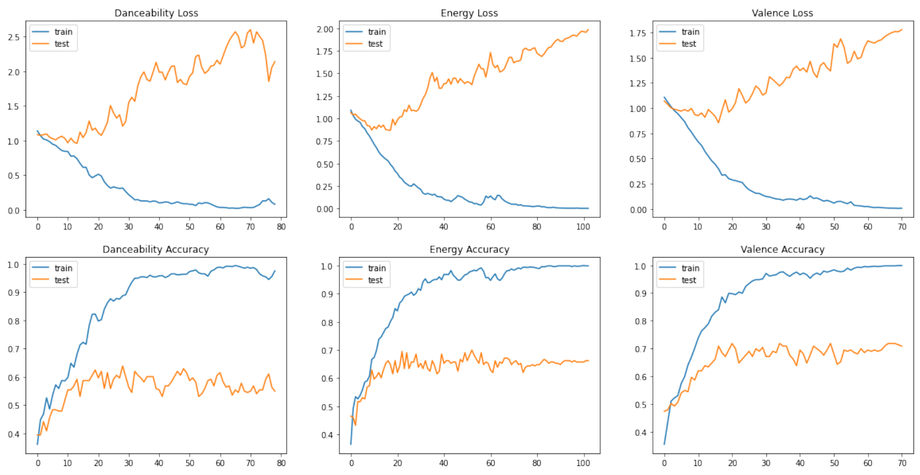

Figure 5.3: Loss and accuracy plots for LSTM model

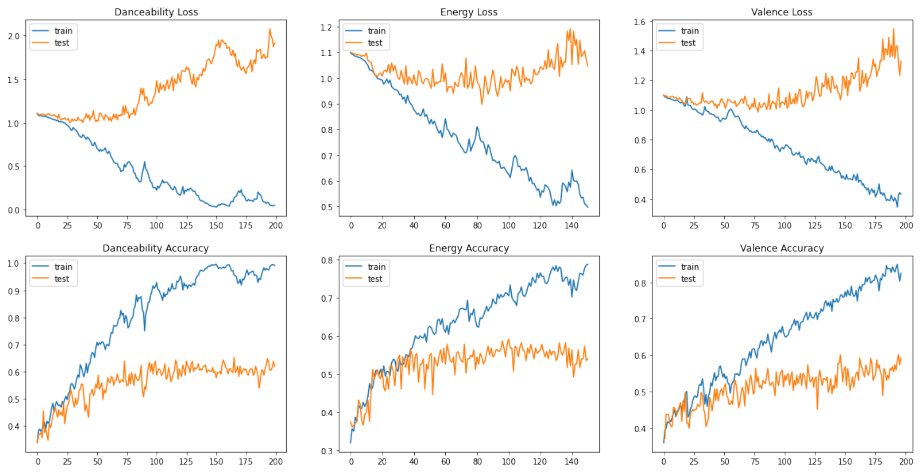

Figure 5.4: Loss and accuracy plots for LSTM (no harmony) model

5.2 Confusion Matrices

The confusion matrices for the RNN and LSTM models show no significant differences in recall and precision scores. This is likely due to the dataset being well balanced. The logistic regression model has larger differences in precision and recall scores, but not major differences. The logistic regression, RNN, and LSTM models show roughly clean confusion matrices with the diagonals being clearly visible. Only the LSTM model trained without harmony and the transformer has less clean confusion matrices. The transformer had particular trouble with valence as it seems the model mostly predicts a valence score of 0.

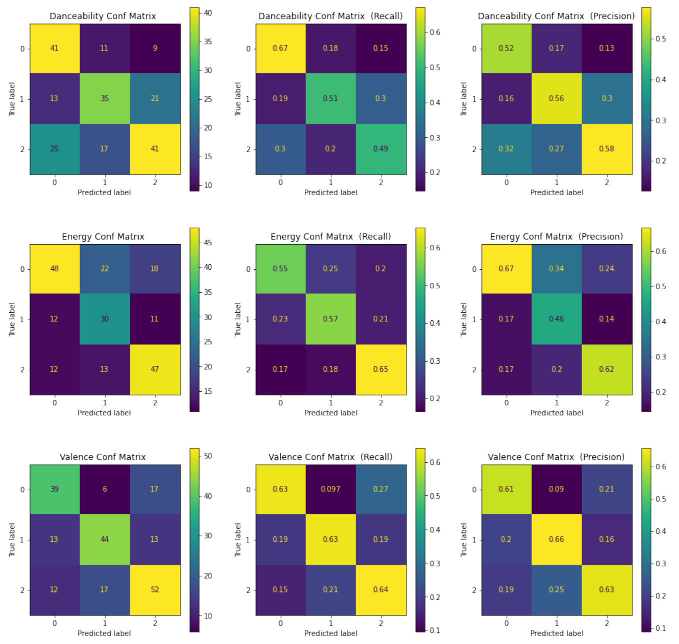

Figure 5.5: Confusion matrices for logistic regression model

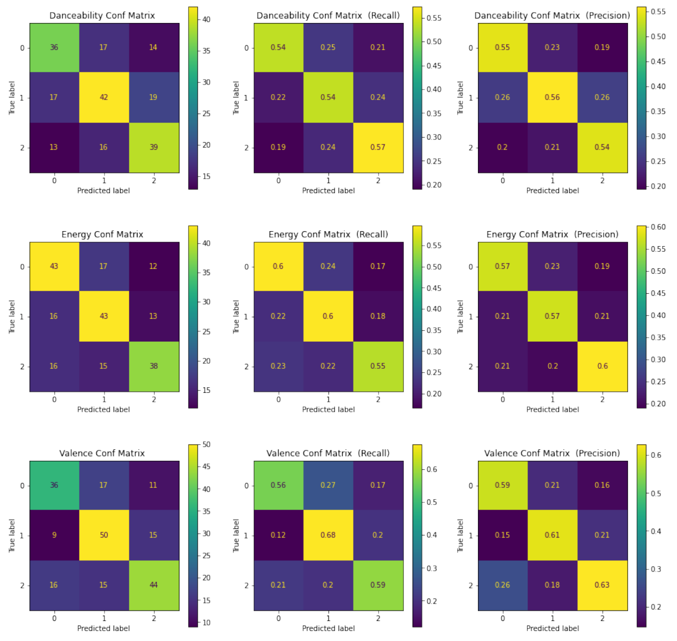

Figure 5.6: Confusion matrices for RNN model

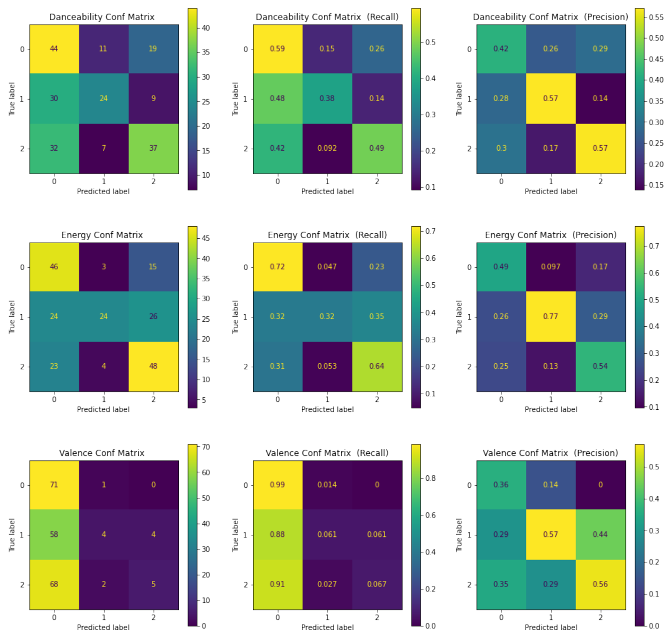
Figure 5.7: Confusion matrices for transformer model.

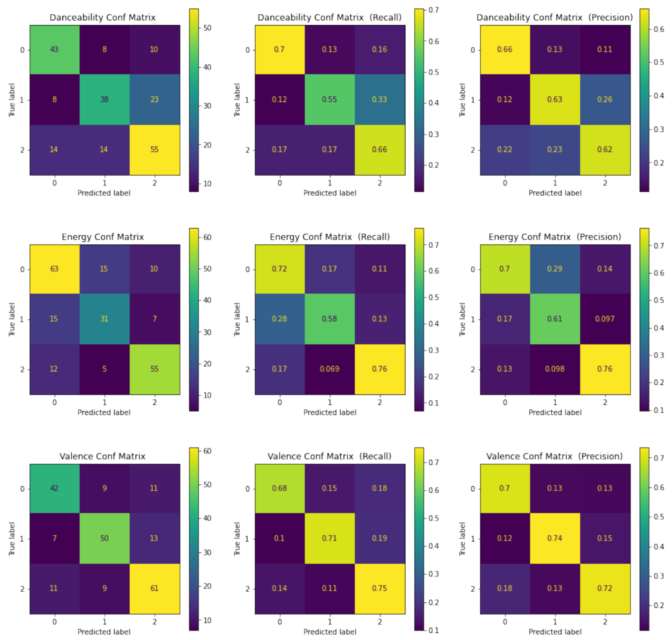
Figure 5.8: Confusion matrices for LSTM model

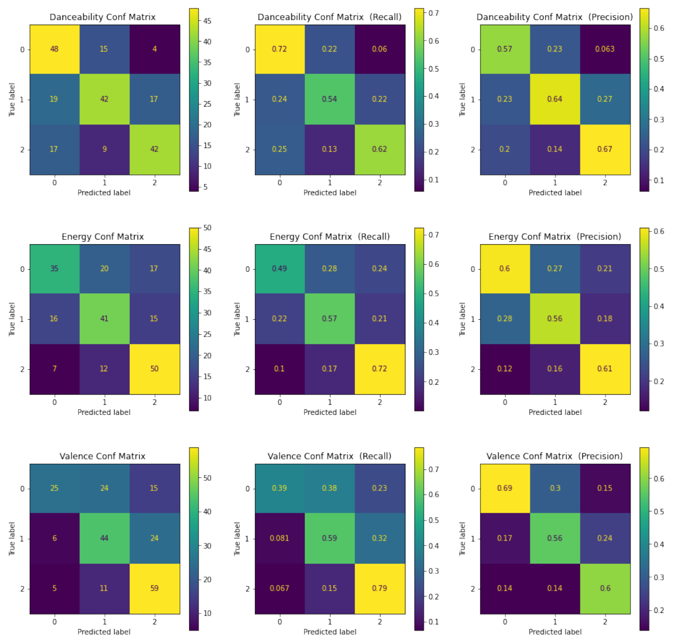

Figure 5.9: Confusion matrices for LSTM (no harmony) model

5.3 Model Accuracies

Because our dataset is roughly balanced, we decided to use accuracy as our metric to analyze our models. After training all of our models, we obtained the following accuracies for each sentiment class (best models highlighted):

Sentiment Class	Model	Accuracy
Danceability	Logistic Regression	54.93%
	RNN	54.93%
	Transformer	49.84%
	LSTM	63.85%
	LSTM (No Harmony)	**65.26%**
Energy	Logistic Regression	58.69%
	RNN	58.22%
	Transformer	52.62%
	LSTM	**69.95%**
	LSTM (No Harmony)	59.15%
Valence	Logistic Regression	63.38%
	RNN	61.03%
	Transformer	56.80 %
	LSTM	**71.83%**
	LSTM (No Harmony)	60.09%

Table 5.1: Accuracies for all models and all sentiment classes

Overall, the LSTM model beats out all other models for each sentiment class. The RNN model surprisingly performs worse than the logistic regression model in each sentiment class despite the fact that RNNs are specifically designed for sequential data. The transformer model, despite its power and expressive potential, also loses to logistic regression — it must be acknowledged that it has significant overfitting potential given the small dataset size.

When training the LSTM without harmony, the danceability accuracy beats all other models, but is close to the regular LSTM's danceability accuracy. The energy accuracy drops compared to the LSTM trained with harmony, but the energy accuracy is still higher than the non-LSTM models. Finally, the valence accuracy drops more than 10% after removing harmony.

6 Discussion

Having run several classes of models, we've come across results both surprising and expected. LSTMs greatly outperform the other models given the same features, though removing harmony leads the LSTM to perform slightly better for danceability and significantly worse for energy and for valence. The logistic regression models also perform surprisingly well relative to RNN. In this section, we provide some possible explanations for the results we received. Our explorations with transformers clearly requires more data and serves more as a proof-of-concept than anything; it is not discussed extensively below.

6.1 Findings

We first compare our logistic regression and plain RNN models. Why might a model built for sequential data be outperformed by logistic regression? Part of the reason could be that RNNs need significantly more data than regression methods to perform well — this dataset, with about a thousand performances, may not be enough for the RNN. Another reason is that plain RNNs frequently suffer from a vanishing gradient problem. In our case, we're working with sequences of 100 notes, and gradients may become too small to influence training during backpropagation.

To solve the vanishing gradient problem, we use the LSTM architecture instead. Not only does this prevent the the aforementioned problem — it's also just a better way to model dependencies between musical notes and chords. In Western classical music, the function or purpose of a single note or chord is highly dependent on those preceding and following it.

For example, a simple C Major chord (stacking the notes C, E, and G) may have many different purposes depending on what key it's in: without getting into the details, in the key of C Major, a C Major chord would be the most defining chord (the tonic or I chord) of the piece; in the key of F minor, its purpose would be the dominant or V chord, which eventually drives to an F minor chord[1]. We'd

[1] A key signature can be thought of as the overall harmonic context of a piece or section of a piece. A key is characterized by certain notes and chords, which may also appear in other

typically associate major or minor keys as happy or sad, respectively (though that's an oversimplification), so understanding the long-term context of a chord also helps us infer the key signature and valence (mood) of a piece.

For the reasons stated above, the LSTM's ability to account for long-term dependencies between notes and chords leads to much better performance than the other baseline models. Note that a bi-directional architecture was consistently chosen through hyperparameter tuning, and this fits with the narrative that musical notes and chords are heavily tied to what follows as well as what precedes them. We can also see that harmony is a big factor used by the LSTM; removing it leads to a huge drop in performance for valence and energy. For danceability, the performance actually increases — this may be because danceability is largely influenced by rhythmic patterns and consistency as well as tempo, all of which may be better inferred using the duration and pitch of notes alone (the use of harmony in this case could be unparsimonious or lead to overfitting).

Our methods involving the bi-LSTM demonstrate a way for computers to "understand" the subjective emotion of a piece of music just by analyzing its notes. In theory, this architecture could be used for any MIDI piano recording, which can be easily generated with digital pianos in one's home as well as certain modern concert pianos. When coupled with other programs which translate audio[2] or music scores to MIDI, we can see that use cases for this method extend beyond digital piano recordings — it simply focuses on the features of music which can be represented through MIDI piano. Some key differences between our method and other programs such as Microsoft's MusicBERT[Zen+21] is the lightweight nature of LSTMs versus transformers, as well as the fact that we do not use information about beats or bars (to know which notes fall in each beat or bar is to align human performances with a musical score; this can be difficult with human-performed classical music since tempo or speed often fluctuates, and notes don't always fall neatly into a grid-like rhythm). Ideas in our methods may be of interest to music streaming platforms who wish to understand their users' listening preferences and recommend music to them. Additionally, contemporary multimedia art drawing upon digital technology and data[3] may find musical sentiment analysis techniques useful.

6.2 Limitations and Ethical Considerations

A primary limitation in this study is the unavailability of human sentiment data for large music datasets. Spotify API data, while convenient and fairly accurate, is largely computationally derived, meaning that our models cannot learn anything about the sentiment of the music that Spotify's models haven't. This being the said, our methods are intended as a framework for future research with

keys; all keys have some major and some minor chords.

[2]`https://www.lunaverus.com/`

[3]https://www.philorch.org/your-philorch/learn-more/Blog/missa-solemnis-2.0-dreaming-beethoven-into-a-new-life/

human labels rather than for direct application — although direct application is possible.

A second limitation comes in the dataset structure and preprocessing methods used. While ”step”, ”pitch”, and ”duration” technically encode most of the information of a note, this representation does leave out much of the information about the performance, such as velocity and beat and bar information. The latter has been implemented in a variety of successful MIDI-based sentiment analysis algorithms, including Microsoft's MusicBERT and MIDI-BERT [Cho+21]. The models in this paper would likely have benefited strongly from this kind of structuring (but such data is often not available, especially the beats and bars of human performances).

A third limitation is the limited size of the training dataset. Eight-hundred-and-fifty 100-note sequences was not enough to fully train the simle transformer model, and it likely wasn't enough to maximize the potential of the LSTM model, either. To augment the size of the dataset, shorter overlapping sequences within this 100-note limit could have been used. To further augment the data, and to prevent overreliance on precise time measurement, ”duration” and ”step” could have been treated with noise. Beyond this, the best solution would simply be to gather a larger group of labelled MIDI files and perform more comprehensive training.

While the ethical considerations of music sentiment analysis are light, accurate sentiment analysis is important as it factors in to the recommender systems used by popular streaming services like Apple Music and Spotify. Faulty recommendations by this kind of service could lead to (1) users not finding music that matches their tastes, and (2) new artists potentially not getting the recognition that they deserve.

6.3 Future Research Directions

Further research in this area could focus on (1) expanding data to include information about the time signatures, tempos, beats and bars, and other embeddings of harmony in the music, (2) expanding the pool of sentiment metrics, to include things like ”liveness” and ”instrumentalness”, (3) using a service like Amazon's Mechanical Turk to collect human-generated labels for sentiment metrics, and (4) designing a hybrid recommender system that takes sentiment analysis into account in addition to nearest-neighbors listening patterns.

7 Conclusions

Overall, we found that the LSTM model was best suited for classifying the sentiment of piano music. This is not particularly surprising since LSTMs are specifically designed for sequential data, and are more advanced versions of RNNs. We also found that using bidirectional LSTM layers helped improve accuracy for all sentiment classes. Adding harmony as a feature to our dataset also helped increase the model accuracies significantly, specifically for the valence and energy sentiment classes. We ultimately believe that our findings can be easily adapted to classify other kinds of music, not just piano music.

References

[Cho+21] Yi-Hui Chou et al. "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding". In: *CoRR* abs/2107.05223 (2021). arXiv: `2107.05223`. URL: `https://arxiv.org/abs/2107.05223`.

[Haw+19] Curtis Hawthorne et al. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset". In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=r1lYRjC9F7`.

[Lam+] Paul Lamere et al. *Spotipy 2.19.0 Documentation.* `https://spotipy.readthedocs.io/en/2.19.0/#`. Accessed: 2022-04-28.

[NS18] Katleen Napier and Lior Shamir. "Quantitative Sentiment Analysis of Lyrics in Popular Music". In: *Journal of Popular Music Studies* 30 (Dec. 2018), pp. 161–176. DOI: `10.1525/jpms.2018.300411`.

[QCZ22] Jibao Qiu, C. L. Philip Chen, and Tong Zhang. "A Novel Multi-Task Learning Method for Symbolic Music Emotion Recognition". In: *CoRR* abs/2201.05782 (2022). arXiv: `2201.05782`. URL: `https://arxiv.org/abs/2201.05782`.

[Zen+21] Mingliang Zeng et al. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training". In: *CoRR* abs/2106.05630 (2021). arXiv: `2106.05630`. URL: `https://arxiv.org/abs/2106.05630`.