A Multi-User Virtual World with Music Recommendations and Mood-Based Virtual Effects

Charats Burch, Robert Sprowl, Mehmet Ergezer

Wentworth Institute of Technology
550 Huntington Ave, Boston, MA 02115
burchc@wit.edu, sprowlr@wit.edu, ergezerm@wit.edu

Abstract

The SEND/RETURN (S/R) project is created to explore the efficacy of content-based music recommendations alongside a uniquely generated Unreal Engine 5 (UE5) virtual environment based on audio features. S/R employs both a k-means clustering algorithm using audio features and a fast pattern matching (FPM) algorithm using 30-second audio signals to find similar-sounding songs to recommend to users. The feature values of the recommended song are then communicated via HTTP to the UE5 virtual environment, which changes a number of effects in real time. All of this is being replicated from a listen-server to other clients to create a multiplayer audio session. S/R successfully creates a lightweight online environment that replicates song information to all clients and suggests new songs that alter the world around you. In this work, we extend S/R by training a convolutional neural network using Mel-spectrograms of 30-second audio samples to predict the mood of a song. This model can then orchestrate the post-processing effect in the UE5 virtual environment. The developed convolutional model had a validation accuracy of 67.5% in predicting 4 moods ('calm','energetic','happy','sad').

Introduction

In the task of music recommendation, mood recognition is a key feature in understanding a piece of music. Previous work has shown a link between personality traits and musical taste (Vuoskoski and Eerola 2011; Rentfrow and Gosling 2003; Zangerle et al. 2018), as well as genre (Ferwerda, Tkalcic, and Schedl 2017), which makes further understanding of psychological factors an important part of music recommender systems (MRS) (Schedl et al. 2018). While the problem often "lacks a well-defined answer" (Kim et al. 2010), the classification of mood in music nonetheless is a popular problem to solve. There are multiple approaches used, such as contextual information, content-based, or hybrid approaches (Kim et al. 2010). This project continues to focus on content-based methods, in order to avoid the common bias of MRS to lean towards more popular content (Ferraro et al. 2021).

The motivation for this project was to explore the efficacy of using Mel-spectrograms in mood recognition of music. Given that the Mel scale is a near-human accurate representation of audio (Stevens, Volkmann, and Newman 1937; Stevens and Volkmann 1940) and has often been used in similar work for music (Venkataramanan and Rajamohan 2019; Hsu, Chen, and Yang 2021), one would expect this to continue into mood classification. This builds off our original work which explores content-based music recommendations, in contrast to traditional user-based recommendations. The overall aim is to find and use the most accurate representation of audio samples to the human ear and anticipate seeing the results reflect that accuracy.

Thayer's model (Thayer 1990) is a popular choice for emotion classification (Bischoff et al. 2009; Singh et al. 2012) and has been used for audio applications such as playlist creation based on mood (Panda and Paiva 2011), artificial emotions forecasting (Salmeron 2012) and song visualization (Kim et al. 2017). Mood prediction has previously been done using valence values, specifically, arousal-valence values (Kim et al. 2011; Delbouys et al. 2018). (Veas 2020) predicts mood with ten of the features provided by Spotify: length, danceability, acousticness, energy, instrumentalness, liveness, valence, loudness, speechiness, and tempo. However, Spotify audio features may not always be readily available. This body of work aims to replicate the accuracy of such models by solely using Mel-spectrograms which may be effective in discerning music genres/sub-genres (Hsu, Chen, and Yang 2021) as well as emotion in speech audio (Venkataramanan and Rajamohan 2019).

Our project's contribution is threefold: First, we created a scalable virtual world for multiple users to experience music together. This environment is generated using Unreal Engine 5 (UE5) with environmental effects and actors which are determined by the currently playing song's audio features. We then provided song recommendations based on a fast pattern matching (FPM) algorithm. Lastly, we developed a convolutional model to control the environment's post-processing effect, based on the mood of the song being played.

SEND/RETURN Overview

The motivation of this project was to both improve existing music recommendation algorithms and the song searching experience. We wanted to create a new and unique way for people to get together and experience new music; to create a relaxed space to explore music and visually experience the

difference between tracks, which is different from the way we usually perceive new music.

This project solves both these issues, with a sophisticated, yet robust, song recommendation algorithm and an experiential multiplayer virtual experience. This allows users to find new music alongside their friends wherever they are in the world. This project also creates a unique way to experience songs visually. One may even view songs they have listened to in the past in a new light once they become aware of the environmental changes that affect different songs.

Network Overview

SEND/RETURN (S/R) is comprised of three main components: A Flask server, a virtual world executable, and a Python script to generate new dataset entries using the Spotify API. All three components work together to create the uniquely generated experience powered by Unreal Engine 5. Figure 1 illustrates the S/R server/client model created.

A Spotify scraper was created in Python that allows us to gather data on a per-album basis. The data we gather includes 30-second previews for each song and a compiled list of all audio features for each track in CSV format. Using the Spotify scraper increased flexibility and was used to quickly expand the dataset with a compiled list of album names, and automate any data wrangling processes required to use the data. The compiled information that is received from the script is used to train our models and used as a reference for key information within the Flask application for hosts. More information about how we created the dataset is presented in the following section. The original S/R project relies on the 'valence' feature provided by Spotify to control the mood in the generated virtual environment. We propose a convolutional model to predict the mood of each song and replace this audio feature.

The Unreal Engine 5 multiplayer game framework powers the front-end of S/R by allowing a user to host a listen-server for other people to join into. Within the listen-server, all data is replicated using the game framework using Remote Procedure Calls and client multicast replication. This increases the scalability of the project as the host is the only user that needs to set up and interact with the Flask server. Using replication on our particle and post-processing systems allows clients to simply download the executable for the virtual world and join listen-servers using a lobby list browser or a direct IP connection. From the host's system, an input allows for song searching, and song submission data is sent to the local Flask server in which our model returns a JSON object that contains values that will be replicated to all connected clients (Grinberg 2018). This is our lightweight solution for S/R that makes it easy to jump in in and share the discovery of new music with others.

The Flask server acts as an interface that handles data between our model and the virtual world that we built. Specifically, Flask is used to map our model to API endpoints that are called in Unreal using VaREST, an HTTP plugin. The Flask application is designed to receive a string from the Listen-server host and uses the Spotify API to find the correct 30-second preview for analysis. Our trained STUMPY model will return a song name from our dataset and Flask

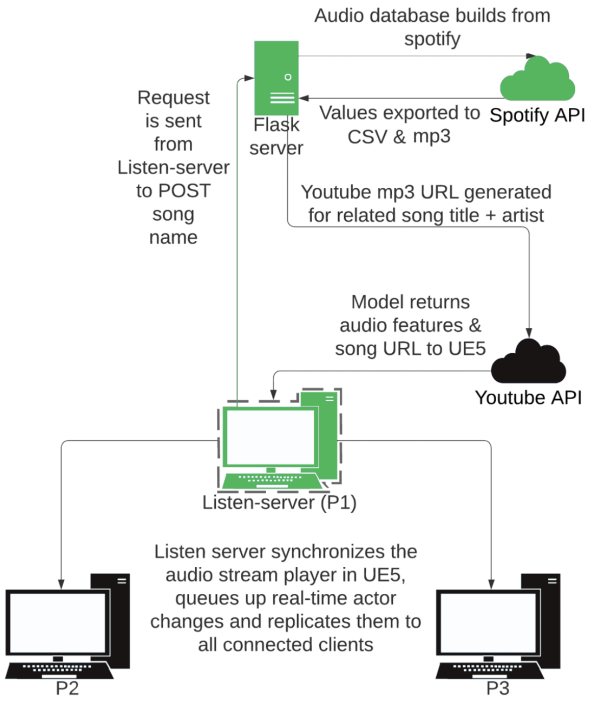


Figure 1: SEND/RETURN network diagram.

will utilize the YouTube API to find a direct link to the song on YouTube.

Data returned from Flask is packaged up in a JSON format that allows Unreal Engine 5 to parse the information into the game world assets. With the returned object, the 'streamingurl' parameter is set server-side in an audio streaming asset found in the world and replicated to all connected clients using multicasting. This creates a replicated audio stream that all connected players can hear with audio occlusion and reverb based on the environmental location in the virtual world. Other data packaged in the returned JSON object refers to the Spotify audio feature values for the returned song, with the values being used to modify multiple systems replicated from host to client, as outlined in Table 1. The mapping of audio features to environmental effects was based on experimentation by the project's designers. We experimented in order to find the visual effect, which this model dictates, that has the largest impact on human experience. Examples of these effects in the virtual world are shown in Figure 2 and Figure 3 respectively.

With all of these systems in place, S/R successfully creates a lightweight online environment that replicates song information to all clients and suggests new songs that alter the world around you.

Song Recommendation

Song recommendation in S/R supports two algorithms: k-means clustering of audio features and STUMPY's (Law 2019) fast pattern matching (FPM) of audio signals. The input song can be any song that is searchable in Spotify's API, and the algorithms return a song from our created dataset.

Figure 2: Example of S/R environment effect changes based on audio features. The falling fireworks are associated with a low-energy song.

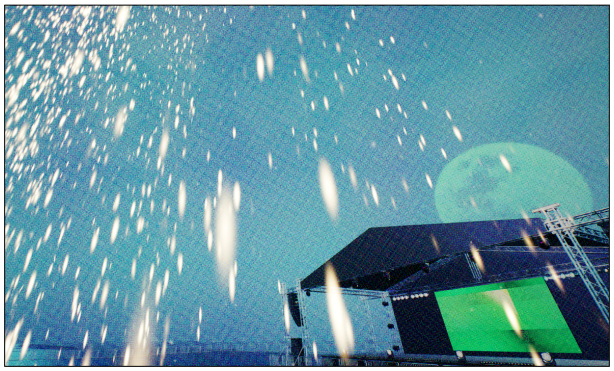


Figure 3: Example of S/R environment effect changes based on audio features. The bright blue post-processing effect is associated with a high-tempo song.

Audio Feature	Environmental Effect	Data Type
Mode	Moon Color	bool (m,M)
Tempo	Post-Processing Effect	int (bpm)
Time Signature	Blueprint Shader	int (3 to 7)
Valence	Player Model/Darkness	float (0 to 1)
Energy	Firework Projectile	float (0 to 1)
Danceability	Meteor/Firework Count	float (0 to 1)

Table 1: Unreal Engine environmental effects and their corresponding audio features. Mode's data type is either minor (m) or major (M).

Note that k-means clustering is the only algorithm that relies on audio features provided by Spotify.

Both k-means and FPM are clustering algorithms that employ unlabeled data and aim to separate songs into n groups of similar within group statistics. The k-means utilizes Lloyd's algorithm to find centroids that minimizes within-cluster sum of squares (Lloyd 2019). FPM is a clustering algorithm that is designed to efficiently work on time series data. It relies on calculating the matrix profile to compute the distances between time-series signals using Mueen's Algorithm for Similarity Search (Mueen et al. 2022).

The k-means clustering algorithm runs much quicker ($\sim$5 seconds, including communication) than the FPM (up to $\sim$30 seconds, including communication). However, based on our qualitative feedback from student demos, FPM far outperforms k-means clustering when it comes to users feeling the recommended song is similar to their input song. For this reason, we focused on developing the FPM algorithm over k-means and set FPM as the default choice in the app.

The k-means clustering algorithm clustered all the songs in our dataset randomly into 50-250 clusters and was trained on the Spotify audio features of each song in our dataset. It then found the nearest cluster for the input song, based on its audio features, and randomly chose a song from the nearest cluster. The reliance on randomness of this algorithm is likely one of the reasons it failed to recommend similar sounding songs, according to user feedback.

In contrast, FPM utilizes the audio signal (in lieu of audio features generated by Spotify) to find the input song's closest matching pattern(s) in our dataset. In order to keep recommendation times below 30 seconds, the audio signals were down-sampled before pattern matching. Another limitation of this approach is that the algorithm searches through the entire dataset of songs, which means recommendation time would increase as the dataset size increases. FPM uses matrix profiles to reduce the time complexity to O(log(n)), but it may eventually run into run-time issues even if we move our algorithm to a system with higher specifications. While 30-second computation time is higher, we generally have about three minutes to select the next song and could reduce computation times further with caching.

Since FPM proved more successful with pairwise matching, we tested pairwise k-means clustering and found it to perform worse than larger numbered clusters.

During a project showcase at our University, we set up a demo station with VR glasses and collected qualitative feedback from approximately ten users. The users donned the headset, searched for a song of their choosing and our algorithm provided a similar song from our dataset and altered the virtual environment accordingly. All of the users who tested our project demo preferred the recommendations generated by FPM over k-means. Most of the limitations of our recommendation came when our dataset did not include a similar song to match a user's search preferences. For instance, if a user chose a genre not defined in our dataset, we wouldn't be able to recommend a similar song. Overall, the users found the experience pleasant.

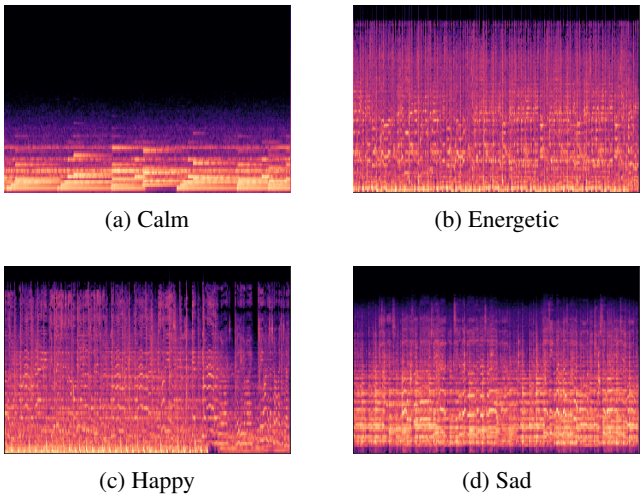
Dataset

To create a dataset to train our mood-predicting neural network to detect moods, we collected the top five playlists for each of the four moods using Spotify's API. We then converted the 30-second audio previews to Mel-scaled spectrograms. We prefer this scaling due to its perceptual-centric design where scale of pitches are judged by listeners to be equal in distance one from another. Algorithm 1 outlines our steps for creating the dataset. Note that we chose the keyword 'relax' to generate 'calm' songs as there was an issue with 'calm' playlists not consistently returning playlists

Algorithm 1: Dataset collection

Input: Moods
Output: Mel-scaled spectrogram

Moods = 'happy','sad','energetic','relax'
for each mood: m **do**
    Query Spotify for 5 playlists for the US market for m
end for
for each song in the playlists: s **do**
    Request 30-second preview of each s
    Compute short-time Fourier transform (STFT)
    Save the magnitude from the complex-valued STFT
    Calculate their Mel-scaled spectrogram
    Convert to dB-scaled spectrogram
end for
return dB-scaled spectrograms



(a) Calm (b) Energetic

(c) Happy (d) Sad

Figure 4: Sample song spectograms

were in the top results of a user searching 'calm' on Spotify. We found that 'relax' was more successful in returning 'calm' playlists. While each playlist can be a different length, they are about 75 to 100 songs each. We ended up with 1,292 total songs where each song generated a spectrogram that is 128 by 2584 pixels. A sample of 225 spectrograms is randomly chosen from each of the four target moods in our dataset. The resulting set is split 700/200 (0.78/0.22) for train/validation. A sample spectrogram generated for each mood is shown in Figure 4. Note that these figures are randomly selected and not representative of the whole dataset.

All the features in our dataset, listed in Table 1, are normalized to avoid features like tempo to dominate others.

Methods

Our objective is to train a model that employs audio features to generate virtual environmental effects. We observe the significance of these visual changes when a model is able to successfully distinguish between calm vs energetic and sad vs happy tunes. Thus, we aim to not just maximize our model accuracy but also to develop a model that provides a pleasant virtual environment to our users.

Algorithm 2: Convolutional model design. The 2D convolution layers are represented as "conv (kernel size for the convolution window) - (number of output filters)". The ReLU activations are not shown for brevity. Fully-connected layers are listed as "fc - (number of neurons)"

Input: Mel-scaled spectrogram (128,2584)
Output: Predicted mood class

conv2-32 + batch normalization + max pooling
conv2-64 + batch normalization + max pooling
conv2-128 + batch normalization + max pooling
conv2-256 + batch normalization + max pooling
fc-256 + batch normalization + 50% drop-out
fc-128 + batch normalization + 50% drop-out
fc-64 + batch normalization + 50% drop-out
fc-4 with softmax activation
return Mood $\in$ {'happy','sad','energetic','relax'}

Model Design

The employed model is a 2D convolutional network as visualized in Figure 5 (Gavrikov 2020). It consists of 2D convolution layers with 32, 64, 128, and 256 filters, respectively, each followed by batch normalization and a max pooling operation. Before the head of the network, there are densely connected layers with 256, 128, and 64 neurons each with batch normalization and 50% dropout rate and ReLU activation. Since we are aiming to categorize each song into four moods, the output layer has four neurons with a softmax activation. The model is presented in Algorithm 2.

As an alternative to building our own model, we also experimented with fine-tuning existing image classifiers, such as VGG16 (Simonyan and Zisserman 2014). But their overall accuracy was not comparable to the proposed model.

Model Training

The model is trained for 50 epochs to minimize sparse categorical cross-entropy loss using rmsprop (Hinton, Srivastava, and Swersky 2012). The train and validation are shown in Figure 6. While the results indicate possible overfitting occurring after around 45 epochs, we enabled Keras's early stopping callback to save the best model which occurred at epoch 41.

Results

S/R provides a virtual environment for a group of users to co-listen to music. The UE5 environment is uniquely augmented for the currently playing song, based on its Spotify audio features. An FPM algorithm is employed to recommend users a similar sounding song based on the raw audio signals.

The proposed CNN model dictates one of the environmental effects, in place of Spotify's audio feature 'valence'. The following are quantitative results for the CNN model which predicts a song's mood from among the four of

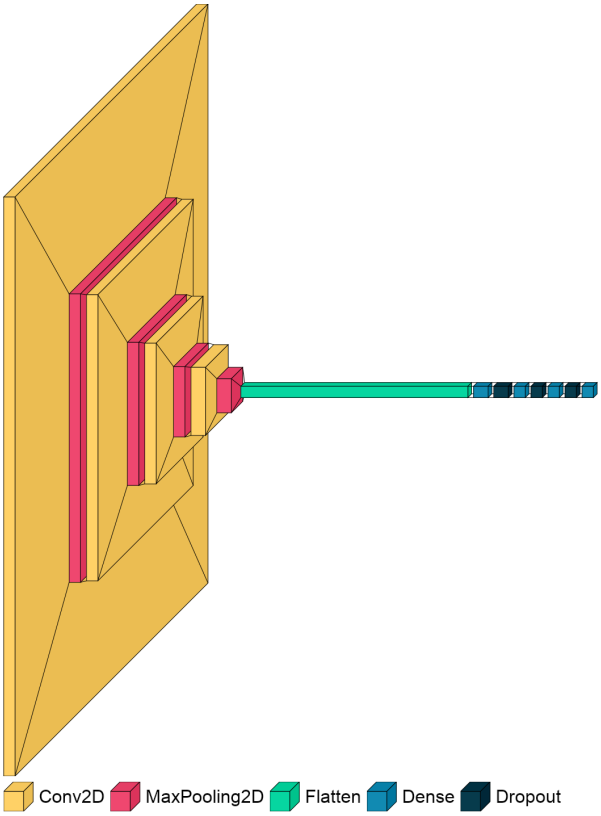


Figure 5: Convolutional model architecture

Mood	Precision	Recall
Calm	0.98	0.72
Energetic	0.55	0.55
Happy	0.57	0.56
Sad	0.65	0.87

Table 2: Precision/recall scores for the predicted moods.

Thayer's 2D emotion model. The highest validation accuracy for the model is 67.5%, with reruns consistently scoring around 65%. A confusion matrix of our predicted moods is shown in Figure 7 and associated precision/recall scores are provided in Table 2.

We found the environment post-processing to be the most significant effect on user experience as it affects the entire screen hue at all times. The Spotify API may recommend 'valence' to reflect the mood, but we propose to control this via the mood prediction CNN model instead.

Our model performs the strongest identifying 'calm' spectrograms, with a precision of 0.98, as they are the most distinct ones as shown in Figure 4. Following this visible pattern, 'sad' has the highest recall of 0.87. This performance is lessened when it comes to 'happy' and 'energetic' spectrograms; the model seems to struggle with these two moods, typically confusing them almost 40% of the time. Looking at our other trained models, the models seem to generally learn 'sad' and 'calm' well (recalls > 0.70, precision > 0.65), and

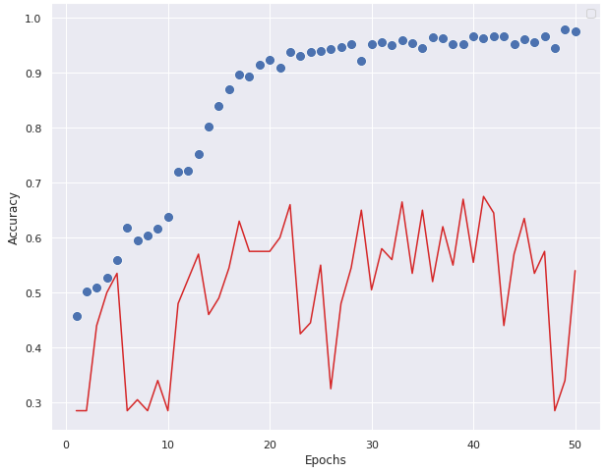


Figure 6: History of the train (blue dots) and validation (red line) accuracy.

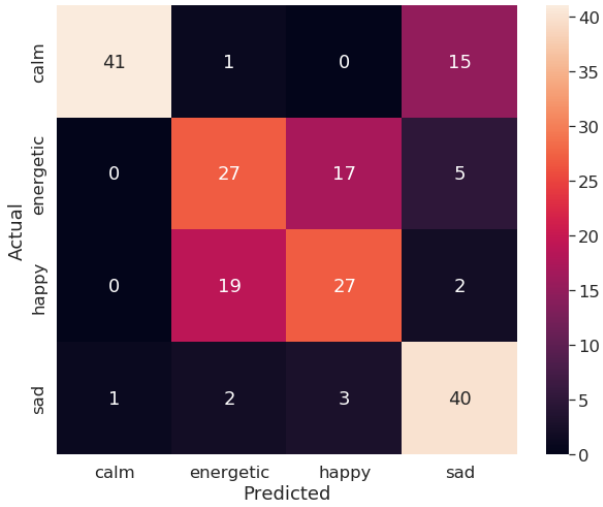


Figure 7: Confusion matrix of the model with accuracy: 0.675

'energetic' and 'happy' somewhat well (recalls and precisions around 0.55 each).

While a higher accuracy may be needed for certain applications, we believe the overlap of happy and energetic songs is acceptable for our use case of controlling effects in a virtual environment. Therefore, we believe these results are adequate to replace Spotify's 'valence' audio feature with the caveat that the 'energetic' and 'happy' labels be combined given their ambiguity in our results. This is an improvement from employing Spotify's 'valence' value as frequently it could not differentiate between 'sad' and 'calm'.

Conclusion and Future Work

This paper presented a virtual world that provides a scalable solution for multiple people to experience new songs together. SEND/RETURN creates a world where many en-

vironmental factors are controlled by the streaming music's features. It includes a song recommendation algorithm based on fast pattern matching to explore new music and a mood detection model that alters the virtual effects in their environment.

A convolutional neural network was employed for mood recognition/classification, achieving a 67.5% validation accuracy for four-class (calm, energetic, happy, sad) classification when using Mel-spectrograms as the sole input feature.

The model is especially effective at differentiating between calm vs energetic and sad vs happy songs. While the accuracy score of the model may be improved with a larger dataset and/or data augmentation, which other models make use of, it has been adequate to control the mood in the virtual environment.

Data augmentation and the building of a larger dataset could give the model more breadth of learning and flexibility. The model struggled to learn for a while in the development of this project but adding a BatchNormalization layer after each Conv2D layer solved this issue. Thus, experimenting with other layers or architectures, such as a short-chunk CNN with ResNet (Hsu, Chen, and Yang 2021), can help with learning. We also would like to explore multimodal approaches, i.e. incorporating lyrical mood classification (Delbouys et al. 2018; Hu, Downie, and Ehmann 2009). Given the effectiveness of Mel-spectrogams in representing music, we would like to explore using Mel-spectrograms as the backbone of a song recommendation model, likely a Siamese neural network (Manocha et al. 2018; Lee et al. 2020).

Currently, the mapping of audio features to environmental effects is designed manually. In future, a separate study can be conducted to develop an AI model to control these environmental effects. This model could rely on audio features as well as external indicators such as crowd patterns or even weather forecast.

The choice of song recommendation algorithm was based on a qualitative feedback collected from approximately ten users. Both the environmental effects and song recommendation could be enhanced by a larger, more comprehensive user study to show the impact of AI on human interaction with music.

References

Bischoff, K.; Firan, C. S.; Paiu, R.; Nejdl, W.; Laurier, C.; and Sordo, M. 2009. Music mood and theme classification-a hybrid approach. In *ISMIR*, 657–662.

Delbouys, R.; Hennequin, R.; Piccoli, F.; Royo-Letelier, J.; and Moussallam, M. 2018. Music mood detection based on audio and lyrics with deep neural net. *arXiv preprint arXiv:1809.07276*.

Ferraro, A.; et al. 2021. *Music recommender systems: taking into account the artists' perspective*. Ph.D. thesis, Universitat Pompeu Fabra.

Ferwerda, B.; Tkalcic, M.; and Schedl, M. 2017. Personality traits and music genres: What do people prefer to listen to? In *Proceedings of the 25th conference on user modeling, adaptation and personalization*, 285–288.

Gavrikov, P. 2020. visualkeras. https://github.com/paulgavrikov/visualkeras. Accessed: 2022-08-31.

Grinberg, M. 2018. *Flask web development: developing web applications with Python*. " O'Reilly Media, Inc.".

Hinton, G.; Srivastava, N.; and Swersky, K. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8): 2.

Hsu, W.-H.; Chen, B.-Y.; and Yang, Y.-H. 2021. Deep learning based EDM subgenre classification using Mel-spectrogram and tempogram features. *arXiv preprint arXiv:2110.08862*.

Hu, X.; Downie, J. S.; and Ehmann, A. F. 2009. Lyric text mining in music mood classification. *American music*, 183(5,049): 2–209.

Kim, D.; Van Ho, P.; Lim, Y.; et al. 2017. A new recognition method for visualizing music emotion. *International Journal of Electrical and Computer Engineering*, 7(3): 1246.

Kim, J.; Lee, S.; Kim, S.; and Yoo, W. Y. 2011. Music mood classification model based on arousal-valence values. In *13th International Conference on Advanced Communication Technology (ICACT2011)*, 292–295. IEEE.

Kim, Y. E.; Schmidt, E. M.; Migneco, R.; Morton, B. G.; Richardson, P.; Scott, J.; Speck, J. A.; and Turnbull, D. 2010. Music emotion recognition: A state of the art review. In *Proc. ISMIR*, volume 86, 937–952.

Law, S. M. 2019. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39): 1504.

Lee, J.; Bryan, N. J.; Salamon, J.; Jin, Z.; and Nam, J. 2020. Disentangled multidimensional metric learning for music similarity. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6–10. IEEE.

Lloyd, S. P. 2019. Least squares quantization in PCM; 1982. *Search in*.

Manocha, P.; Badlani, R.; Kumar, A.; Shah, A.; Elizalde, B.; and Raj, B. 2018. Content-based representations of audio using Siamese neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3136–3140. IEEE.

Mueen, A.; Zhing, S.; Zhu, Y.; Yeh, M.; Kamgar, K.; Viswanathan, K.; Gupta, C.; and Keogh, E. 2022. The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance and Correlation Coefficient. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html. Accessed: 2022-08-31.

Panda, R.; and Paiva, R. P. 2011. Automatic creation of mood playlists in the Thayer plane: A methodology and a comparative study. In *8th Sound and Music Computing Conference (SMC2011)*.

Rentfrow, P. J.; and Gosling, S. D. 2003. The do re mi's of everyday life: the structure and personality correlates of music preferences. *Journal of personality and social psychology*, 84(6): 1236.

Salmeron, J. L. 2012. Fuzzy cognitive maps for artificial emotions forecasting. *Applied Soft Computing*, 12(12): 3704–3710.

Schedl, M.; Zamani, H.; Chen, C.-W.; Deldjoo, Y.; and Elahi, M. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2): 95–116.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, P.; Kapoor, A.; Kaushik, V.; and Maringanti, H. B. 2012. Architecture for automated tagging and clustering of song files according to mood. *arXiv preprint arXiv:1206.2484*.

Stevens, S. S.; and Volkmann, J. 1940. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3): 329–353.

Stevens, S. S.; Volkmann, J.; and Newman, E. 1937. The mel scale equates the magnitude of perceived differences in pitch at different frequencies. *Journal of the Acoustical Society of America*, 8(3): 185–190.

Thayer, R. E. 1990. *The Biopsychology of Mood and Arousal*. Oxford University Press.

Veas, C. 2020. Predicting the music mood of a song with deep learning. https://github.com/cristobalvch/Spotify-Machine-Learning. Accessed: 2022-08-31.

Venkataramanan, K.; and Rajamohan, H. R. 2019. Emotion recognition from speech. *arXiv preprint arXiv:1912.10458*.

Vuoskoski, J. K.; and Eerola, T. 2011. The role of mood and personality in the perception of emotions represented by music. *Cortex*, 47(9): 1099–1106.

Zangerle, E.; Chen, C.-M.; Tsai, M.-F.; and Yang, Y.-H. 2018. Leveraging affective hashtags for ranking music recommendations. *IEEE Transactions on Affective Computing*, 12(1): 78–91.