MUSIC RECOGNITION

VANGUR EBENEZER RAHUL DEEPAK - B200818CS
KANCHIREDDY VARSHITHREDDY - B200832CS
NAREDLA ROHITHREDDY - B200794CS

Under the guidance of

MANJUSHA K
Assistant Professor- CSED
National Institute of Technology Calicut

***Abstract*- Our project aims to develop a music recognition system capable of identifying songs even when they are played in the background, sung by ordinary individuals without instrumental accompaniment, or hummed by someone. current music retrieval systems retrieve music by searching artists, phrases in the lyrics, and so on but most users can hum songs that they want to search but there is no widely deployed way that could accomplish this. We aim to develop a system which could do this.**

I INTRODUCTION

In the contemporary era, search engine technology has become an indispensable tool for information retrieval. The majority of search queries revolve around words or images, effectively serving the needs of most requests. However, for music searches, these common query types often fall short. Frequently, individuals have a melody stuck in their minds, known as an "earworm," making traditional searches inadequate. so a natural solution called "query by humming," where humming the desired song is the most effective approach for a music search. The difficulty and frustration of being unable to describe a melody using words or images creates a need for a different, natural method of music search. This irritation is perhaps what inspired academics to investigate a strategy that mimics a basic and inborn behaviour—humming or singing a melody.

" People frequently and naturally hum songs to express a melody because it allows them to share the aural feelings that the music arouses. This need is intended to be met by the "query by humming" initiative, which makes use of people's natural ability to hum or sing a melody as a query. The goal is to bridge the gap between the natural and digital worlds of music, making it possible to efficiently and intuitively search for music by directly expressing the tune that comes to mind. In an age where music plays an increasingly important part in our lives, the project aims to improve the user experience and make music retrieval more accessible, accurate, and user-friendly. To comprehend the content of the paper, it's essential to grasp

the following fundamental terms:

1.**Pitch:** Pitch refers to the perception of a sound wave's frequency, determining whether the sound is high or low. Higher frequencies create a higher pitch, while lower frequencies result in a lower pitch.

2.**Loudness:** Loudness pertains to the volume or intensity of a sound as perceived by the human ear. It is influenced by the sound wave's amplitude, with greater amplitudes producing louder sounds and lesser amplitudes yield softer ones.

3.**Treble:** Treble describes higher-frequency sounds in music, primarily associated with the upper range of musical tones. It is fundamental in musical compositions, providing a sense of brightness and clarity to the overall sound.

4.**Notes:** In music, notes serve as the fundamental units of sound, representing specific pitches corresponding to particular frequencies. Notes are denoted by letters from A to G, with variations indicated by Sharp: (#) , Flat : (♭)

5.**Melody:** A melody is a sequence of musical notes played or sung consecutively, creating a recognizable and often memorable musical line. Melodies are central to musical compositions, serving as the primary tune or theme that listeners can easily follow and remember.

6.**Spectrogram:** A spectrogram is a visual depiction of a sound or musical piece's frequency spectrum. It shows how the intensity of different frequencies changes over time, offering a detailed view of the sound's frequency composition.

II LITERATURE SURVEY

[6] This approach can detect songs when they are played in the background in the presence of noise. They first convert the audio to a spectrogram. To sparse the data they selected the top N peaks(N being proportional to the length of the song). Then we identify peaks within the songs using a maximum filter. And then they form pairs of points and store each pair in the database i.e. key being $(f1, f2, \text{time\_diff})$.

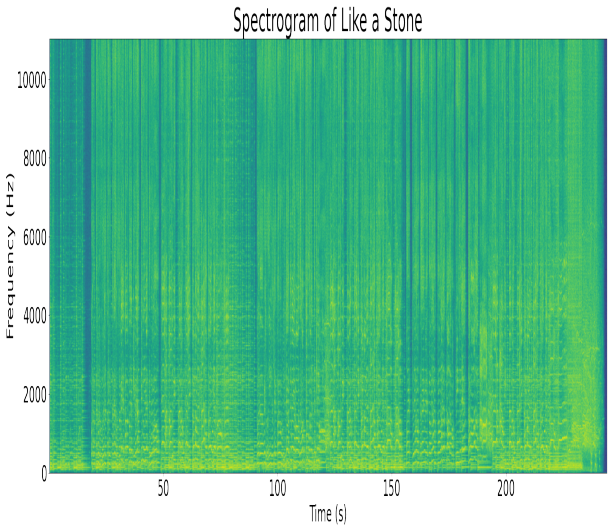


Figure 1: Spectrogram (x: time, y: frequency)

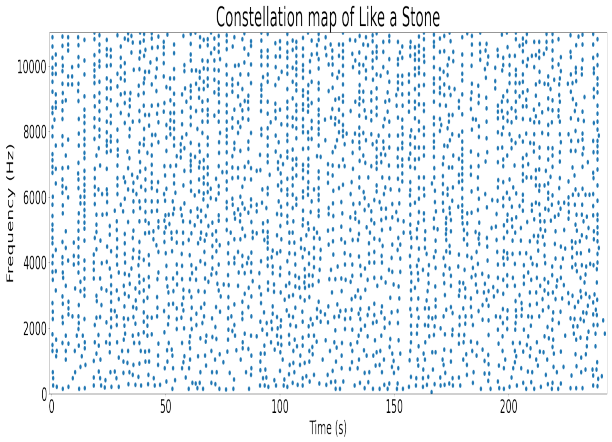


Figure 2: peaks (x: time, y: frequency)

This procedure relies on the fact that even in the presence of noise the peaks are preserved. And also saving the pairs of frequencies along

with the time difference between them instead of saving each point and their absolute time allows us to search the song efficiently and allows the query to be from any arbitrary section of the song.

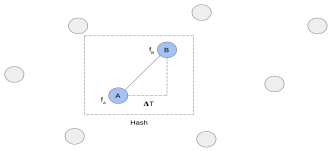


Figure 3: pairing up peaks

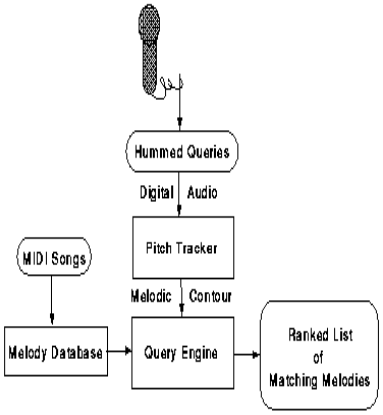


Figure 4: system Architecture

Drawbacks: The major drawback of this approach is it can only recognize the song when it's played in the background as it uses the fact $(f1, f2, \text{time\_diff})$. is invariant but this is not the case when sung or hummed by a user as they may sing it at a slightly different pace. And may Sing it at a different pitch. And thereby $(f1, f2, \text{time\_diff})$. won't be the same. And there is no guarantee that the peaks would not all be from the instruments thereby not queryable by humming.

[2] This approach relies on the fact that we distinguish two melodies by their melodic counter (defined as the relative pitch difference). They transform the audio into a sequence of pitch transitions i.e. S referring to the Same as the previous, D lower than the previous, and U higher than the previous so we transform our query to get a sequence of characters and hand it over to search. The main components of this approach are the pitch tracker and search. The pitch tracker here uses a technique called autocorrelation to generate the sequence. And then they employ a fuzzy search algorithm to find the best match.

Drawbacks: The main problem with this technique is it doesn't scale well with this fuzzy search algorithm over a large set of songs. and their pitch tracker is slow as it takes about 45 seconds to pitch track for 10 seconds of 44KHz, 16-bit audio. And for songs they have used MIDI files not the original song as pitch detection from original songs is quite challenging and cannot be accomplished using the above technique because of multiple sources of sound (i.e. instruments and singers).

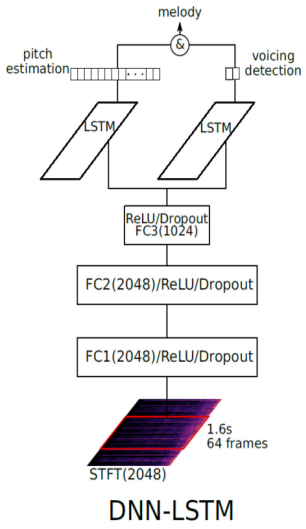
[1] This paper modifies the [2] paper based on their observation of the fact that people tend to hum the song or remember the melody of the sung that is sung by the singer thus they conclude that indexing the song with UDS descriptor (i.e encoding the song as U, S, D as in [2]) only on the voiced part of the original song so they extract the voice from the original song and then index the song. They also employ a somewhat more efficient search algorithm.

Drawbacks: This basically cannot work on OSTs and instrumentals. And the voice extraction isn't mentioned properly so assuming they indexed the original voiced version of the song. And there is also the fact that for some songs the melody is in the instruments so it won't work if the hummed melody is a part of the instruments.

[4] [3] These papers use DNN-LSTM to generate a more complex descriptor of the

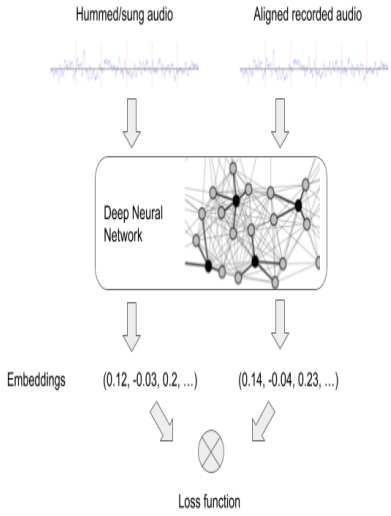
music. They have used a multitasking model that learns on two tasks: pitch estimation and voicing (i.e., presence of melody or not) detection. thereby being able to work even on a polyphonic complex musical recording.

The pitch estimation part and the voicing detection part are both classification neural networks. The voicing detection neural network classifies to 1, 0 indicating the presence of melody or not and the pitch-estimation neural network classifies pitch into 540 classes (i.e. their reasoning behind this approach is humans being sensitive to frequencies in the range 55Hz to 1.76 kHz with 1/9 semitone resolution). After the generation of the sequence, they convert the sequence into a sequence of relative differences in the original sequence and they employ a unified search that uses an n-gram inverted index to search in RP4G, RP3G, and RP2G. and if it fails they use an MNF normalization technique to the query and then search.



DNN-LSTM

Drawbacks: Assuming that the resolution capability of humans and reproducibility capability are of the same resolution (when deciding the number of pitch classes). The training of this neural network also requires sufficiently high-quality data. [5] Here the idea is they take two versions of the same song i.e. original song and hum and generate the same embedding for both versions of the song. They employ a triple loss function. The search is performed as the closest vector to the embedding. But they haven't quite mentioned the exact model that they used and also how they dealt with the fact that the query can be from any arbitrary section of the song.



III MOTIVATION

Imagine wanting to find a song, but you don't remember the words or the name of the song. You just have the tune, that catchy melody playing in your mind. Regular search methods, like typing words into a search engine, don't work in this situation. This is where "query by humming" becomes important. It's a way for you to find that song by humming or singing the tune you have in your head.

When we hum or sing a melody, we're expressing how the music feels to us. It's a natural way for us to share that musical feeling with others. "Query by humming" takes advantage of this natural behaviour. Instead of using words, you can hum the tune of the song you're thinking of, and the system

will try to find the song that matches that tune. It's like a bridge between our natural way of expressing music and the digital world of finding songs online.

In today's world, music is a big part of our lives. We hear songs everywhere, and they often stick with us. So, making it easier for us to find and enjoy the songs we love is a great goal. "Query by humming" helps us do just that. It makes finding music more accessible and fun, especially when we have that tune playing in our heads but can't put it into words.

IV Design

Our Design consists of 2 major components. First component is meant to retrieve the song when it's played in the background using the method described in [6]. The second component is responsible for retrieving the song when the query is sung or hummed by a person.

The Design of the first component is mostly the same as described in [6]. But the design of the second component involves two Neural Networks

fingerprint in component1: the audio fingerprints represents combination of peak finding and the hashing of peaks within a range to a particular peak , where peaks represents loudest frequency at a particular point.It is represented by 5-tuple (freqA, freqB, time-diff, star-time, song-id)

fingerprint in component2: they represent the UDS descriptor sequence, where ('U', 'D', 'S') represents the pitch of a note is above, below, same , than the pitch of the previous note.

Threshold in component1: we use a certain score to set the result in the component 1, using which we will be able to get the correct result where the value of the threshold isn't fixed, where we plan to fic a value while we do experimenting.

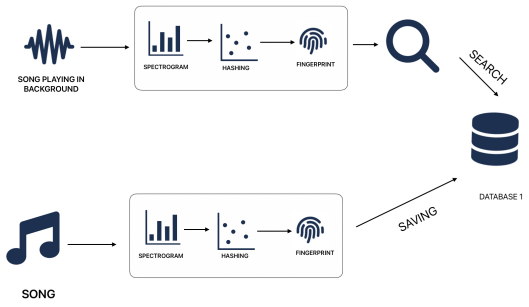


Figure 5: component-1

Neural networks in component2:

1. Hum/MIDI to UDS (N1)
2. Original Song to UDS (N2)

N1 Hum/MIDI to UDS: This neural network takes in the input of two successive frames (i.e. a small time sample of audio) of audio-spectrogram it outputs a descriptor for that instance. Then we do this for the entire song to get a UDS annotation of that Hum/MIDI.

N2 Original Song to UDS: This neural network takes in the original song as the input same as N1, we split the song into small time frames and generate a UDS annotation of that song.

To train this neural network we have taken the MIDI version of the original song and generated a UDS annotation using N1 and used it as the expected output of N2 for the song.

So whenever we have a query the query is fed to the first component then if it finds a match with a score greater than a threshold

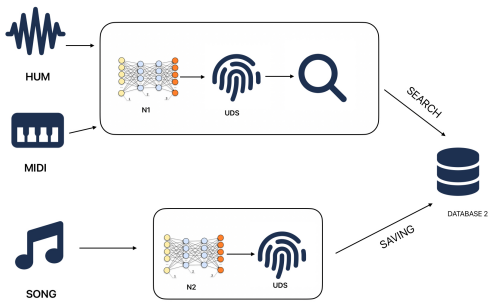


Figure 6: component-2

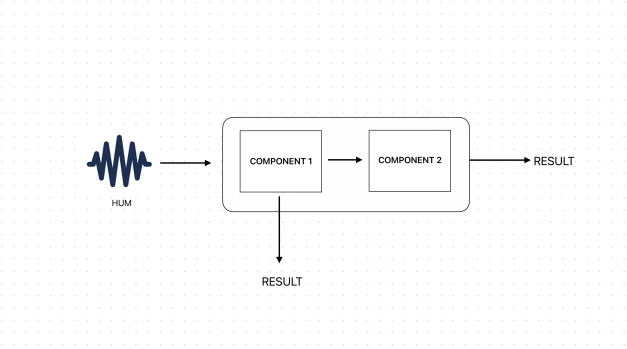


Figure 7: program

value we return the match. In case it doesn't have enough score, we feed it into the second component then we return the match from here.

The first and second components store two different fingerprints of the song. i.e. The first stores the fingerprint as a set of 5-tuples (freqA, freqB, time-diff, star-time, song-id). It stores this as a key-value pair where the first 3 are keys and the other two are values. whereas the second component stores the fingerprint as a UDS annotation and we employ a fuzzy search for this database.

V Experiments

we initially thought of using neural networks, but we will also try DNN-LSTM , and some other neural networks for better accuracy and results.

neural network N1 and N2: we are going to feed the frames of the song , where the number of frames is to be decided while experimenting.initially we thought of feeding two frames which are current and the previous . we are also going to set the activation function, weights to the nodes and number of nodes and also the number of classifications other than ('U','D','S') to be evaluated while experimenting using backpropogation.

Search :

Hashing in component1 :

1.Retrieve all hashes from the database that matches the sample fingerprint and group these hashes by song 2.for each song , figure out if the hashes matches. 3.choose the track with most matched hashes.

Fuzzy search : The closeness of a match is measured in terms of the number of primitive operations necessary to convert the string into an exact match. This number is called the edit distance between the string and the pattern. The usual primitive operations are: insertion, deletion, substitution.

we calculate the similarity using the formulae ,

$$\text{similarity} = 1 - \frac{\text{edit\_distance}}{\min(\text{len}(\text{term}), \text{len}(\text{word}))} \quad (1)$$

where edit_distance is the min number of perations done to equate the strings and term is the length of the generated string of hum and word is the length of the substring of the song which we took for comparision.

VI Evaluation metrics

we use mean reciprocal rank for measuring the accuracy of the model.The mean reciprocal rank is a statistic measure for evaluating any process that produces a list of possible

responses to a sample of queries, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer: 1 for first place, 1/2 for second place, 1/3 for third place and so on. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}.$$

where rank_i refers to the rank position of the first relevant document for the i-th query.

The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

VII CONCLUSION

Music recognition system helps users who can hum the melody but cannot recall the artist or song name benefit from our work, which involves the user providing input to the system in the form of a hum, and our technology identifies the song for him, making the procedure very simple.

Deep learning techniques, such as neural networks and LSTM, are increasingly being used in music recognition, according to the literature. The integration enables the extraction of the melody using various techniques and recognition of the patterns of the query using various search algorithms with the original songs in the database, and the most appropriate songs being returned as the output.

References

[1] Edwin Alfaro-Paredes, Leonardo Alfaro-Carrasco, and Willy Ugarte. Query by humming for song identification using voice isolation. In *Advances and Trends in Artificial Intelligence. From Theory to Practice: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part II*, page 323–334, Berlin, Heidelberg, 2021. Springer-Verlag.

[2] Yingyi Bu, Raymond Chi-Wing Wong, and Ada Wai-Chee Fu. Query by humming. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 2244–2249, Boston, MA, 2009. Springer US.

[3] Zhengyu Cao, Xiangyi Feng, and Wei Li. A multi-task learning approach for melody extraction. In Haifeng Li, Shengchen Li, Lin Ma, Chunying Fang, and Yidan Zhu, editors, *Proceedings of the 7th Conference on Sound and Music Technology (CSMT)*, pages 53–65, Singapore, 2020. Springer Singapore.

[4] Andreas Novian Dwi Triastanto and Rila Mandala. Query by humming music information retrieval using dnn-lstm based melody extraction and noise filtration. In *2022 5th International Conference on Information and Communications Technology (ICOIACT)*, pages 503–508, 2022.

[5] Chrisian Frank. The machine learning behind hum to search. 2020.

[6] Avery Wang. An industrial strength audio search algorithm. In *International Society for Music Information Retrieval Conference*, 2003.