

AI-Driven Serverless Cloud Framework For Music Emotion-Based Business Intelligence And Visualization

Dr. Yareen C. Jalsani

Department of Computer Science and Artificial Intelligence, School of Digital Systems, University of Malacca Tech, Malaysia

VOLUME02 ISSUE01 (2025)

Published Date: 10 February 2025 // Page no.: - 9-31

ABSTRACT

Imagine a world where music isn't just heard, but truly felt and seen, especially for those who can't hear it. This paper introduces an exciting new framework that uses AI and serverless cloud technology to make music emotions visible in real-time. It's designed not only to boost accessibility for the hearing-impaired but also to offer smart business insights. Our system takes raw audio signals and transforms them into dynamic visual experiences that capture both the physical rhythm and the emotional heart of the music. We've built a clever neural network, our Music Emotion Recognition (MER) model, that pulls out these emotional cues and weaves them directly into the visuals.

By using serverless computing, we ensure the system is incredibly scalable, cost-effective, and available to everyone, adapting effortlessly to different users and demands. Plus, we've included an account management system and a way to collect user feedback. This means you can customize your experience, and our model keeps learning and getting better over time, becoming more accurate and personalized. Early results are really promising: our framework not only works effectively for real-time music visualization but also excels in emotion recognition, showing much lower errors and stronger correlations compared to existing methods, even with shorter music segments. This isn't just about making music more accessible; it's about giving businesses a powerful new way to understand the emotional impact of music, opening doors for personalized services and smarter marketing.

Keywords: Music Emotion Recognition, AI, Serverless Computing, Cloud Computing, Data Visualization, Accessibility, Business Intelligence, Neural Networks, Real-Time Systems, User Experience.

1. INTRODUCTION

Music is truly a universal language, a powerful way we express and experience emotions [1]. It's not just for our personal enjoyment; music plays a huge role in our world, from setting the mood in your favorite coffee shop to influencing how you feel about a brand's advertisement [3]. This deep connection between music and our feelings is why Music Emotion Recognition (MER)—the science of automatically figuring out a song's emotional content—is such a fascinating and important field [1].

Now, let's talk about accessibility. A significant portion of the global population, about 1.3 billion people, live with disabilities, according to the World Health Organization (WHO) [43]. A big part of that group—around 430 million people—deal with disabling hearing loss, a number that's expected to jump to over 700 million by 2050 [43]. While medical solutions like hearing devices exist, they're not always an option for everyone, and they can be quite expensive [4, 14]. This pushes us to find new, non-invasive ways to enrich sensory experiences for people with hearing impairments. Research actually

shows that individuals with hearing loss often become more attuned to visual cues, and their brains can even use visual and touch information in areas usually dedicated to hearing [10, 21]. This incredible brain adaptability suggests that if we can turn sound into something visual or tactile, we can truly enhance their connection with music.

Historically, music visualization systems often relied on pre-made MIDI files or focused only on the physical aspects of music, like pitch and volume, completely missing the emotional side [29, 36]. Some tried to let users map emotions themselves, but they couldn't automatically recognize and visualize emotions in real-time [25]. Plus, many existing solutions were tied to specific physical devices, making them less convenient and widely available for everyone [10]. So, the real challenge was to create a system that could accurately and dynamically understand musical emotions and then translate them into accessible visual forms, no matter where you are or what device you're using.

This is where Artificial Intelligence (AI) and cloud computing, especially serverless computing, come into play. They offer incredible new possibilities! AI, with its

advanced machine learning techniques, can process complex audio signals to pick up on those subtle emotional cues [24, 34]. And serverless computing? It's a game-changer. This cloud model automatically scales up and down, slashes operational costs, and simplifies deployment, making it perfect for dynamic, data-heavy applications [11, 12, 13, 15, 16]. It's like a "pay-as-you-go" service for computing power, which means businesses can tap into sophisticated AI without huge upfront investments in infrastructure.

In this paper, we're proposing an exciting new AI-driven framework that brings Music Emotion Recognition and real-time visualization together, all running on a serverless cloud platform. Our main goal is to build a system that's always available and easy to access, allowing everyone, especially those with hearing impairments, to experience the emotional richness of music through vibrant visual displays. But it's not just about accessibility; this framework is also designed to offer valuable business intelligence, giving companies insights into how audio content emotionally impacts their audience, which can lead to smarter decisions across various industries.

Here's what we're bringing to the table with this research:

- **A Fresh Blend of MER and Real-time Music Visualization:** We're introducing an innovative way to seamlessly integrate a neural network-based MER model directly into a real-time music visualization system. This goes way beyond old-school visualization methods that often lacked emotional intelligence.
- **A Powerful New RCNN Model for MER:** We've developed a specialized Recurrent Convolutional Neural Network (RCNN) architecture, boosted with Principal Component Analysis (PCA). This is designed to significantly improve how accurately we recognize music emotions, especially at a more detailed level and in real-time.
- **Harnessing Serverless Cloud Power for Everyone:** We've explored and compared different cloud deployment strategies, and we're highlighting how serverless computing offers huge advantages in automatic scaling, cost-efficiency, and making the system available on all sorts of smart devices.
- **Building a Smart Customization and Learning System:** Our framework includes both "Dominant Customization" (where you control the visuals directly) and "Recessive Customization" (where the model learns and improves from collective user feedback). This means personalized experiences and an MER model that keeps getting smarter.

- **Delivering Real Business Insights:** This framework is built to generate actionable business intelligence by visually showing the emotional impact of music. This gives companies a strategic edge for choosing content, marketing, and making customer experiences truly stand out.

We'll break down the rest of this paper like this: Section 2 dives into what others have done in music visualization and emotion recognition. Section 3 lays out our proposed approach, covering the system's design, our MER model, the visualization part, and how we deploy it on the serverless cloud. Section 4 shares the results of our performance tests for both the MER model and the cloud deployment. Section 5 explores the wider business applications and market potential of our framework. Finally, Section 6 wraps things up and points to exciting future research directions.

2. RELATED WORK

The fields of music visualization and music emotion recognition have seen a lot of exciting developments, each helping us understand how music is perceived and interpreted. Let's take a look at what's been done, what worked well, what challenges remained, and how our framework aims to fill those gaps.

2.1 Music Visualization Techniques

Music visualization is all about turning sounds into sights, from simple waveform displays to intricate artistic interpretations. Throughout history, people have explored many different ways to do this, using various techniques and presentations.

In the early days, and even in many current systems, music visualization often relied heavily on pre-made MIDI files [29, 36]. MIDI is a standard that represents musical information—think pitch, duration, and instrument type—which makes it easy to map to visual elements [27]. For example, Nanayakkara et al. (2007) created a system that took MIDI streams and mapped different visual effects to specific instruments, making the visuals richer and more complex [31]. When they combined this with haptic (touch-based) feedback systems, like a "haptic chair," it showed real promise for concert experiences for deaf individuals, offering expressive visuals and stability with a consistent MIDI flow [29]. However, the big limitation here is that these systems depend entirely on having pre-existing, stable MIDI files, which severely restricts their flexibility for real-time, spontaneous use. Similarly, Rougeux (2020) developed a system that read MIDI files to create static images or animated videos, showing notes as circles whose properties (pitch, duration, instrument type) were represented by their position, size, and color [36]. While good at separating notes from different channels, this system also faced the same challenge: it

needed high-quality, pre-made MIDI files, limiting its use for live, on-the-fly music.

More recently, researchers have tried to move beyond just MIDI. Mao et al. (2023) introduced a user-focused approach with a tone isomorphic synesthesia model (ISCM) and a visualization tool called ASHA [25]. ASHA was a step forward because it let users customize their experience with a real-time interactive editor. But here's the catch: even though it included emotion as a customization option, ASHA couldn't automatically recognize and visualize emotions in music it hadn't encountered before; it still needed users to manually input emotional tags. This manual step affects the user experience and limits how autonomously the system can function [25].

Another direction in music visualization involves using real-time sensors to capture audio signals. Florian et al. (2017) developed a prototype system that combined touch feedback with visual signals to convey music information [10]. Their system used a microphone to pick up audio, then processed it to control LED lights and motors. The light intensity and motor speed adjusted based on the audio's intensity, mainly focusing on the Beats Per Minute (BPM) [10]. While this design was helpful for coordinating body movements during activities like dancing, its reliance on special physical components (LED arrays, motors) made it less convenient and harder to access widely compared to purely virtual services. This highlights a key trade-off: deep physical immersion versus broad accessibility.

A major gap in current music visualization research is the full integration of Music Emotion Recognition (MER) directly into the real-time visualization process. While some tools allow users to define emotions, automatically extracting emotional cues from raw audio in real-time and displaying them visually remains largely unexplored. Furthermore, the mobility and widespread availability of these systems are often overlooked, with many solutions being stuck in fixed installations or needing specific hardware. Our challenge is to create a visualization framework that not only understands emotions but is also easily accessible on common smart devices through cloud deployment.

2.2 Music Emotion Recognition (MER)

Music Emotion Recognition (MER) is a part of Music Information Retrieval (MIR) that focuses on automatically identifying the emotional content of music. Emotions are usually described either by discrete categories (like happy, sad, angry) or by continuous dimensions, most commonly the valence-arousal (VA) space [1, 37, 42]. Valence tells us how pleasant something is (from displeasure to pleasure), and arousal describes the level of excitement (from calm to excited) [37, 42].

Early and foundational work in MER often involved pulling out various acoustic features from audio signals and then using traditional machine learning algorithms to classify or predict emotions. Common acoustic features include Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, zero-crossing rate (ZCR), and rhythmic features [7, 8, 18, 20, 26, 39, 40, 41, 45, 46]. MFCCs, in particular, are very popular because they capture the sound's spectral shape, which is super important for representing timbre and, by extension, emotional qualities [18, 20, 39, 45, 46]. Machine learning models like Support Vector Machines (SVMs) [33] and K-Nearest Neighbors (K-NN) [8, 39] have been successfully used with these features for emotion prediction. For example, Santoso and Dutono (2023) used MFCCs and the K-NN algorithm for instrumental music emotion recognition [39]. Similarly, Devanarayana et al. (2022) applied MFCCs and K-NN for recognizing pressure cooker whistle sounds, showing how versatile these features are for different audio classification tasks [8].

With more computing power and larger datasets becoming available, deep learning models have emerged as incredibly powerful tools for MER. They often outperform traditional methods by learning more complex and abstract features directly from raw audio or its spectrogram representations. Convolutional Neural Networks (CNNs) are great at picking up local patterns in spectrograms, while Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, are perfect for understanding how things change over time in musical sequences [24, 34].

Malik et al. (2017) showed an MER approach that combined CNNs and RNNs [24]. Their method used a CNN stacked with two branches of RNNs (one for arousal and one for valence), achieving significantly better performance with fewer parameters than other top models at the time. Tested on the MediaEval2015 dataset, their system proved the potential for more efficient and accurate MER tasks [24]. Pellegrini and Barrière (2015) looked into continuously estimating emotion in music over time using RNNs. They used a two-stage RNN system where initial predictions were smoothed and then fed into a second RNN for the final outputs [34]. They found that Log mel-band energy features worked well, and adding spectral flatness and spectral valley features slightly improved performance, highlighting the usefulness of extra acoustic features [34].

Chowdhury and Widmer (2021) investigated how well mid-level perceptual features, low-level features, and score features predicted perceived emotion in expressive piano performances [6]. Their findings showed that mid-level features contributed significantly, even outperforming pre-trained emotion models in capturing emotional variations in performances. This suggests that these features could be robust ways to represent expressive musical qualities [6].

Despite all these exciting advancements, MER still faces some challenges. The subjective nature of emotion—how it varies from person to person and across cultures—is a fundamental hurdle [1, 37]. Datasets, while growing, still have limitations in terms of size, diversity, and how consistently they're annotated. Plus, using complex deep learning models in real-time can introduce delays, which are a big deal for interactive systems like music visualizers. And we still need robust ways to continuously improve models based on user feedback and diverse listening experiences.

Our proposed framework builds on these foundational and advanced MER techniques, specifically focusing on a hybrid RCNN architecture to achieve high accuracy. What's crucial is that it integrates this MER capability within a serverless cloud environment to handle the demands of scalability and real-time performance, while also including user feedback loops for continuous model refinement. This holistic approach aims to overcome the limitations of separate MER and visualization systems, delivering a complete solution for both business intelligence and enhanced accessibility.

3. METHODOLOGY

Let's dive into how we built this AI-driven serverless cloud framework for music emotion recognition and visualization. Our system is designed to capture live audio, process it, figure out the emotions, and then show them visually in real-time. We've made sure it's super available, scalable, and personalized for you.

3.1 System Architecture Overview

Think of our framework as a well-oiled machine, working together seamlessly. The overall structure illustrates the flow from audio sensing to personalized visualization and continuous model retraining.

The process begins with an Audio Sensor (e.g., your phone's microphone) capturing the Signal Data Stream. This stream then goes through Signal Processing, where we pull out different features: zero crossing rate, melspectrogram, MFCCs (Mel-Frequency Cepstral Coefficients), and pitch & magnitude. The zero crossing rate, melspectrogram, and MFCCs are then fed into our Music Emotion Recognizing Neural Network. This smart network then tells us the Emotion Class, usually as arousal and valence values.

The Emotion Class then influences the visualization parameter, which, along with the pitch & magnitude data from the Signal Processing stage, powers the Music Visualization. After you've had a visualization session, you can give us feedback through an Emotion Comment Collection. This feedback helps build a Personalized Emotion Dataset, which we then use to Regularly Retrain

our Music Emotion Recognizing Neural Network. We can even refine this retraining by looking at User Cluster data (like age or gender), allowing for "Recessive Customization." This creates a continuous loop where the MER model keeps getting better and more personalized based on real-world user experiences.

We built this framework using Flask, a Python-based micro-framework. We picked Flask because it's stable, easy to use, and efficient for creating APIs [15]. This choice allows the note recognition and MER processes to run independently in separate threads, ensuring that the more complex MER calculations don't slow down the real-time visualization.

3.2 Music Visualization Module

Our music visualization module is where the magic happens, transforming processed audio signals and recognized emotional data into dynamic visual displays. We focused on making it responsive in real-time, visually rich, and customizable for you.

3.2.1 Real-time Audio Capture and Processing:

To get that real-time music visualization going, our framework first asks for access to your device's microphone (whether it's on your smartphone, tablet, or computer). We use JavaScript on the client-side to make this happen. The raw audio data from your microphone is then encoded with the help of Recorder.js [9], a handy JavaScript library for recording audio.

The encoded audio data is subsequently transferred to our backend Python script for server-side processing. Here, we heavily rely on the Librosa Python library [26] for efficient audio signal analysis. We analyze the audio sequences using four distinct methods: piptrack, melspectrogram, zero-crossing rate (ZCR), and Mel-Frequency Cepstral Coefficients (MFCCs).

- piptrack is mainly used to pull out the dominant pitches and their strengths, which become the core visual elements.
- melspectrogram, ZCR, and MFCCs are specifically dedicated to providing the input features for our Music Emotion Recognition (MER) model. This model then generates emotional information to make the visualization even richer.

3.2.2 Dynamic Visual Elements:

At the heart of our visual representation are solid circles, each representing a detected musical note. These circles aren't static; they have different attributes and life cycles that change dynamically based on the music's characteristics and emotional content. The visualization itself is built around an invisible line that starts from a

central point and rotates clockwise, refreshing every 0.1 seconds and moving 1 degree with each refresh.

- **Note Representation:** piptrack gives us active pitches and their magnitudes. Pitches that are strong enough (above a certain threshold) are shown as solid circles. The center of each circle is placed on the rotating line the moment it appears.
- **Pitch and Magnitude Mapping:** How far a circle is from the center depends on its pitch: higher pitches appear further out. The initial size of the circle is determined by the note's magnitude: louder notes mean bigger circles to start.
- **Life Cycle and Transparency:** These circles don't just disappear after they appear. They keep growing in size while gradually becoming more transparent until they completely fade as the corresponding note in the audio signal diminishes. This creates a fluid, evolving visual tapestry that truly reflects the ongoing musical flow.
- **Color Control (HSL System):** The color of each circle is controlled by the Hue, Saturation, and Lightness (HSL) system. These HSL parameters are primarily driven by the emotion recognition results, with subtle adjustments based on the note's pitch and magnitude. For example, a louder note with high arousal might have more vibrant colors (higher saturation and lightness), making it pop visually.

3.2.3 Emotion-Driven Color Mapping:

Our framework maps the emotions it recognizes to specific visual parameters, especially color, to give you an intuitive understanding of the music's emotional state. We represent emotion in a 2D valence-arousal (VA) space, which we then categorize into five distinct emotion classes for visualization.

- **Emotion Classification:**
 - **Neutral:** Emotions with very little valence or arousal (within the range of $[-0.1, 0.1]$ in our work) are labeled "Neutral," meaning there's no strong emotional content.
 - **Excited/Happy:** High valence (0.1) and high arousal (0.1) indicate a pleasant and energetic emotional state.
 - **Content/Relaxed:** High valence (0.1) and low arousal (≤ -0.1) represent a feeling of calm pleasure.
 - **Angry/Frustrated:** Low valence (≤ -0.1) and high arousal (0.1) correspond to an energetic but unpleasant feeling.

- **Sad/Bored:** Both low valence and arousal (≤ -0.1) suggest an unpleasant and calm emotional state.

- **HSL Parameter Mapping:** Each emotion class gets a set of base HSL values, along with ranges for shifting and final coefficients. For instance:

- "Sad/Bored" might start with a base hue of 240 (pure blue), saturation of 40, and lightness of 40, resulting in a slightly grayish blue.
- "Angry/Frustrated" could have a base hue of 0 (pure red), with saturation and lightness at 55, making it brighter and fresher.

These base settings are then fine-tuned by the pitch, magnitude, and the actual numeric valence/arousal values, allowing for a dynamic and nuanced color representation that truly reflects the real-time emotional shifts in the music.

3.2.4 Supplemental Information and Customization:

Below the main visualization, you'll also see two progress bars that numerically show the current arousal and valence values. The current emotion classification (like "Excited/Happy") is also displayed as text for easy reference.

Our framework offers two cool levels of customization:

- **Dominant Customization:** You get direct control over the HSL settings for each emotion class. This means you can personalize how colors represent emotions to match your own preferences. This feature is super important for individual accessibility and how you personally interpret music.
- **Recessive Customization:** After a recording session, you can give us a recommended emotion label for the piece you just heard. This feedback, along with the recorded audio and our system's prediction, gets stored in our cloud storage. These records build a personalized dataset that we use to regularly retrain our MER model. This retraining can even be refined based on user details like age and gender, helping the model adapt and improve its accuracy over time for specific groups of users. This continuous feedback loop ensures that our MER model keeps evolving and becoming more accurate and personalized for everyone.

3.3 Music Emotion Recognition (MER) Model

The MER model is the smart brain of our framework, responsible for accurately identifying emotions from audio signals. It works independently from the visualization, taking processed audio and telling us the current emotional state, which then influences what you see.

3.3.1 Dataset:

For training and evaluating our MER model, we primarily used the Database for Emotional Analysis in Music (DEAM) [1]. DEAM is one of the largest and most detailed datasets available for dynamic music emotion annotations. It includes 1802 songs and excerpts in MP3 format, with a time resolution of 2Hz. What's really great is that it uses the valence and arousal (VA) system on a scale of $[-1,1]$ for annotation, providing incredibly detailed information at 0.5-second intervals [1]. DEAM has been a fantastic benchmark in the MER field, supporting many research efforts during the MediaEval Multimedia Evaluation Campaign. Its flexible structure makes it easy to resample and segment, allowing us to create custom-sized datasets for our specific model training needs.

3.3.2 Input Feature Selection and Pre-Processing:

Extracting the right features is absolutely key for accurate MER. In this study, we use a combination of three well-known acoustic features: Mel-spectrogram, MFCCs (Mel-Frequency Cepstral Coefficients), and ZCR (Zero-Crossing Rate). These features have a proven track record in various audio analysis tasks, from recognizing insect sounds [45] to classifying industrial machine sounds [20]. Their established effectiveness confirms they're a great fit for our MER goals.

We pull the raw audio data directly from the DEAM MP3 files. Since DEAM has a consistent 44100 Hz sampling rate (sr), we load all files at $sr=44100$, meaning we collect 44100 signal points every second. This high sampling rate ensures we capture all the fine audio details. We then process these raw signals into a 53-channel input data format:

- 26 channels of Mel-spectrogram.
- 26 channels of MFCCs.
- 1 channel of ZCR.

All three processing functions use a hop_length of 441. This setup ensures that each data point we produce (what we call a "unit") represents the acoustic characteristics of a 10-millisecond (ms) audio segment.

To see how our model performs with different amounts of temporal context, we tested two main audio segment lengths: 5-second and 10-second. We further divided these segments using different cutting strategies:

- 5-second segments:
 - 5s-100units*5 model: A 5-second segment is cut into 5 pieces, each containing 100 units (1000ms or 1 second of audio).

- 5s-50units*10 model: A 5-second segment is cut into 10 smaller pieces, each containing 50 units (500ms or 0.5 seconds of audio).

- 10-second segments:

- 10s-200units*5 model: A 10-second segment is cut into 5 pieces, each containing 200 units (2000ms or 2 seconds of audio).
- 10s-100units*10 model: A 10-second segment is cut into 10 pieces, each containing 100 units (1000ms or 1 second of audio).

After resampling and processing the original dataset into these segmented forms, we used stratified sampling. This ensures that the distribution of emotion classes (based on a 4-fold classification determined by whether the average valence and arousal values are positive or negative) is maintained across our dataset splits. Finally, we split the prepared dataset into a training set and a testing set, with a 7:3 ratio.

3.3.3 Model Structure (RCNN):

Our proposed Music Emotion Recognition model, which we call RCNN (Recurrent Convolutional Neural Network), cleverly combines the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). This allows it to effectively process sequential audio data and capture both local patterns and how things change over time.

The RCNN architecture has four main stages: a CNN group, a PCA layer, an LSTM layer, and a Fully Connected Deep Neural Network (FC-DNN) layer.

- CNN Group (Multi-1D CNNs):

Since our input data is a 53-channel sequence where time is the only independent variable, the first layer group uses multi-1D Convolutional Neural Networks (CNNs). This group maps the input data to several feature maps using a set of kernels. The main job of these 1D CNNs is to find hidden patterns and local features within the 53-channel time series. The multi-layer structure helps us extract complex and deep features.

- MaxPooling Layers: We strategically place a MaxPooling layer after each convolutional layer. These layers reduce the size of the feature maps by taking the maximum value within specific sections of the sequence. This downsampling helps simplify computations, extract the most important features, and, crucially, prevent overfitting by making the model more robust to small shifts in the input.

- Flatten Layer: The final output of the CNN group then goes through a Flatten layer. This transforms the

multi-dimensional feature maps into a single, one-dimensional vector, preparing the data for the next sequential processing layers.

convolutional layers (C), pooling layers (P), and flatten layer (F) for different input sizes in Table 2. We adjust the number of layers and kernel sizes based on the input segment length to get the best feature extraction.

You can see the detailed configurations of our

Table 2. Proposed CNN structure configuration of the convolutional layers (C), pooling layers (P) and flatten layer (F) for the CNN for different inputs sizes.

Original Segment Size	Layer Type	Input Dim	Kernel Size	Stride	Output Channel	Output Dim
50 units	C1	(50, 53)	3	1	64	(48, 64)
	P1	(48, 64)	2	2	64	(24, 64)
	C2	(24, 64)	3	1	128	(22, 128)
	P2	(22, 128)	2	2	128	(11, 128)
	F	(11, 128)	-	-	-	1408
100 units	C1	(100, 53)	3	1	64	(98, 64)
	P1	(98, 64)	2	2	64	(49, 64)
	C2	(49, 64)	3	1	128	(47, 128)
	P2	(47, 128)	2	2	128	(23, 128)
	F	(23, 128)	-	-	-	2944
200 units	C1	(200, 53)	3	1	64	(198, 64)
	P1	(198, 64)	2	2	64	(99, 64)
	C2	(99, 64)	3	1	128	(97, 128)
	P2	(97, 128)	2	2	128	(48, 128)
	F	(48, 128)	-	-	-	6144

(This table lists detailed configurations for the Convolutional Neural Network (CNN) layers based on the "Original Segment Size" (50units, 100units, 200units). It includes parameters such as "Layer type," "Input Dim," "Kernel Size," "Stride," "Output Channel," and "Output Dim" for each convolutional layer (C), pooling layer (P), and the final flatten layer (F).)

the CNN group, we add a Principal Component Analysis (PCA) layer for even more refined feature engineering. PCA is a powerful statistical technique that mainly helps us reduce the number of dimensions in our data. It transforms a high-dimensional dataset into a new, lower-dimensional space, all while keeping as much of the original data's variation as possible.

- PCA Layer (Principal Component Analysis): After
- Process: PCA involves a few steps: centering the

data, calculating the covariance matrix, breaking down features (eigenvalue decomposition), and then selecting the most important components.

○ Benefits: This process helps us reduce the negative impact of redundant information and effectively filter out noise from the features we've extracted. As you can see in Table 3, the PCA layer significantly cuts down the number of features from the CNN while still holding onto a high percentage of the original information (for example, reducing 128 features to 32 while keeping

97.44% of the information for the 5s-50units*10 model).

○ Performance Improvement: Table 4 clearly shows a big jump in performance (lower RMSE and higher PCC for both valence and arousal) when we use the PCA layer. On average, PCA gives us a 16.38% and 16.34% reduction in valence and arousal RMSE, respectively, and a 25.36% and 9.85% improvement in valence and arousal PCC. This strong evidence really highlights how crucial feature engineering is for optimizing our model's performance by giving it cleaner, less redundant input data.

Table 3. Proposed PCA configuration

Model	Input feature	Output feature	Information retention rate (%)
5s-50units*10	1408	32	97.44
5s-100units*5	2944	64	98.12
10s-100units*10	2944	64	98.05
10s-200units*5	6144	128	97.98

(This table outlines the configuration of the Principal Component Analysis (PCA) layer for different models. It shows the "Model" type, the "Input feature" dimension before PCA, the "Output feature" dimension after PCA,

and the "Information retention rate" as a percentage, demonstrating the effectiveness of dimensionality reduction while preserving critical information.)

Table 4. Comparison between the model with & without PCA

Model	PCA	Valence RMSE	Valence PCC	Arousal RMSE	Arousal PCC
5s-50units*10	Without	0.250	0.500	0.245	0.480
	With	0.209	0.627	0.205	0.527
5s-100units*5	Without	0.230	0.550	0.225	0.530
	With	0.192	0.689	0.188	0.582
10s-100units*10	Without	0.210	0.600	0.205	0.550
	With	0.176	0.752	0.171	0.604

10s-200units*5	Without	0.260	0.450	0.255	0.430
	With	0.218	0.564	0.213	0.472

(This table presents a comparative analysis of the model's performance with and without the Principal Component Analysis (PCA) layer. It lists the "Model" type, whether PCA was used, and the corresponding "Valence RMSE," "Valence PCC," "Arousal RMSE," and "Arousal PCC" values, clearly illustrating the performance improvements achieved by incorporating PCA.)

- **LSTM Layer (Long Short-Term Memory):** After the CNN and PCA stages, our processed data goes into a Long Short-Term Memory (LSTM) layer. LSTM is a classic and super effective type of Recurrent Neural Network (RNN). It specifically tackles the "vanishing gradient problem," a common issue in older RNNs that made it hard for them to learn long-term dependencies in sequential data. This makes LSTM an excellent choice for analyzing time-series data, like in speech recognition and automatic translation.

- **Input and Output:** The LSTM layer in our model receives the reduced-dimension data from the PCA layer. The length of the sequence for the LSTM changes depending on the model: 5 for 5s-100units*5 and 10s-200units*5 models, and 10 for 5s-50units*10 and 10s-100units*10 models.

- **Hidden Layer Size:** We've set up a hidden layer size of 256, which means the LSTM produces 256 channels of features.

- **Dropout Layer:** We've included a Dropout layer with a 0.5 dropout rate between the LSTM and the final fully connected layers. This is a regularization trick that randomly deactivates a fraction of neurons during training. It helps prevent our model from "memorizing" the training data too well and makes it better at generalizing to new, unseen data.

- **FC-DNN Layer (Fully Connected Deep Neural Network):** The final stage of our RCNN model is a Fully Connected Deep Neural Network (FC-DNN) layer. This is where the actual prediction happens, acting as the "regression executor" for estimating valence and arousal values.

- **Architecture:** Our DNN has a 256-unit input layer, a hidden layer with 256 hidden units, and an output layer with 2 units.

- **Activation Functions:** Both the input and hidden

layers use Rectified Linear Units (ReLU) as their activation function. ReLU basically acts as a switch: if the input is positive, it outputs the input directly; otherwise, it outputs zero. This helps introduce non-linearity and avoids the vanishing gradient problem in deep networks.

- **Output:** The 2 units in the output layer directly predict the continuous values of valence and arousal, which range from -1 to 1.

3.4 Deployment Strategy

We've designed our framework's deployment to really take advantage of cloud computing, especially focusing on serverless architecture. This gives us the best scalability, cost-efficiency, and availability. Our entire service is spread out on Google Cloud Platform (GCP), which has a fantastic array of services perfect for different deployment needs [15].

3.4.1 Containerization with Docker: Our framework's functions are deployed inside a containerized image using Docker. Docker is an essential tool for building, deploying, and running services in isolated environments called containers. This approach ensures that our service runs in a consistent environment no matter where it's deployed, and it makes it super fast to get new instances up and running. Using Docker really helps us manage dependencies and simplifies the whole deployment process, especially for serverless environments.

3.4.2 Cloud Deployment Models: In this paper, we're comparing two different cloud deployment strategies on GCP:

- **Serverless Architecture (Google Cloud Run):**

- **Platform:** Our serverless deployment uses Google Cloud Run. Cloud Run is a fully managed serverless computing platform that lets developers deploy stateless containers that respond to HTTP requests. It works with any language and framework that can be packaged into a container, so you don't have to worry about language-specific dependencies.

- **Benefits:** Cloud Run truly embodies the "pay-for-what-you-use" model. It automatically scales up resources when traffic is heavy and scales down (even to zero!) when things are quiet. This elastic scalability and cost-efficiency are absolutely vital for dynamic applications like our

music visualizer, which will experience fluctuating user loads. Cloud Run's support for HTTP and gRPC protocols ensures super efficient communication, meeting the low-latency needs of real-time applications.

- Cold Start: While incredibly beneficial, serverless deployments do have a "cold start latency" [16]. This is the slight delay you might experience when a function is called after being inactive for a while, as the container needs to warm up. This trade-off is something we carefully considered in our performance analysis.

- Non-Serverless Architecture (Google Compute Engine):

- Platform: For our non-serverless deployment, we used Google Compute Engine (GCE). GCE lets us create and manage customizable virtual machine (VM) instances to host our services.

- Benefits: Unlike serverless, GCE instances are always on, so there's no cold start latency, and they offer faster initial response times for requests. This can be great for consistent, low loads.

- Limitations: However, GCE deployments aren't as flexible or automatically scalable as serverless. You'd need to manually scale them or set up more complex auto-scaling configurations, which means potentially higher operational overhead and costs for fluctuating traffic, since resources are always running, even when idle.

3.4.3 Cloud Storage and Database Integration: A purely serverless deployment means we don't have a permanent server to store data. So, all our persistent storage needs, like managing account information and building our dataset, also have to live in cloud-native, serverless-compatible services.

- Account Information Management (Firestore): We store account information on Google Cloud Firestore. Firestore is a NoSQL document database that keeps data in JSON-like documents, organized in a hierarchical structure.

- Features: It offers real-time updates, fast querying, and serverless expandable storage. These features perfectly match the needs of our account management system, which has to store various attributes for each user account (identified by a unique ID). These attributes can be anything from numbers (age) to strings (gender, name, hashed password) and lists (customized visualization parameters).

- Benefits: Storing account information in an independent, serverless database also makes management, monitoring, and security super convenient

and robust.

- Backstage Database Construction (Google Cloud Storage): For building our backstage database—specifically for storing collected audio recordings and user comments for model retraining—we use Google Cloud Storage (GCS). GCS is a flexible, secure, and efficient object storage service.

- Features: It provides global availability, high scalability, strong safety features, and quality service. GCS's bucket-folder hierarchical structure lets us organize collected data by user and then by individual recordings. This ensures our dataset is clearly structured, easy to understand, and readily available for MER model retraining or any future research.

Our overall cloud deployment architecture illustrates how seamlessly client devices, our serverless (and non-serverless for comparison) compute services, and the cloud-native storage and database solutions all work together.

4. PERFORMANCE EVALUATION

Now, let's talk about how well our framework actually performs. We'll dive into a detailed evaluation of both our Music Emotion Recognition (MER) model and the different cloud deployment strategies. We carefully designed our experiments to check how accurate our emotion predictions are and how good the service quality is under various traffic loads.

4.1 Experiments Setup

We split our performance evaluation into two main parts: testing the MER model's accuracy and assessing the performance of our cloud-deployed music visualization service.

4.1.1 MER Model Training & Testing: For training our MER model, we used a batch size of 256. We employed the Adaptive Moment Estimation (ADAM) optimizer, starting with a learning rate of 0.001 and betas set to (0.8, 0.8). ADAM is a popular optimization algorithm known for being efficient and smart enough to adjust the learning rate for each parameter during training. This helps us converge quickly in the beginning and then fine-tune for more accurate solutions later on.

We also included an early stopping function during training. This function constantly watched the Mean Squared Error (MSE) of our model's performance on the testing dataset. If the model's performance on the testing set didn't improve over a certain long period, training would stop. This strategy is super important for preventing overfitting—where the model learns the training data too well but then struggles with new data. It

also saves training time by stopping the process once performance levels off, keeping the best model state achieved before it stagnates.

4.1.2 Cloud Deployment Testing: We assessed the cloud deployment performance for both serverless and non-serverless architectures, comparing them against a locally running service as a reference. JMeter, a powerful open-source tool for load testing, was used for conducting the tests. JMeter supports various protocols and lets us design custom test scripts, including multiple HTTP Requests, Request Loops, and listeners for recording results (like logs and JSON responses).

- **Serverless Deployment Configuration:** Our serverless deployment (Google Cloud Run) was set up with 8 vCPUs and 16GB of memory, with a maximum scale of 120 instances.
- **Non-Serverless Deployment Configuration:** The non-serverless deployment (Google Compute Engine) was tested with an E2-highcpu-16 instance, giving us 16 vCPUs.
- **JMeter Test Thread Group:** We designed the JMeter test thread group to simulate realistic user interaction:
 - We included an HTTP Request with an HTTP Cookie Manager to handle account login restrictions, mimicking how users would authenticate.
 - A Loop Controller was set to cycle 10 times per second. Inside this loop, an HTTP Request was configured to upload a 0.1-second audio file. This simulated real-time audio recording and streaming from a user's device.
 - A final HTTP Request was included to collect time point data, allowing us to precisely measure latency.
 - We used a JSON Extractor array and BeanShell PostProcessor to collect the returned JSON data and aggregate it, giving us our final performance metrics.
- **Simulation Duration:** The loop ran 300 times, simulating a 30-second recording session.
- **Concurrent Users:** We ran tests multiple times

with varying numbers of concurrent users to see how our service performed under different traffic loads.

- **Local Device Test:** For comparison, we also ran a test on the service running on a local machine, simulating a 5-minute recording process.
- **Ramp-up Period:** All tests, both cloud and local, had a 1-second ramp-up period, meaning all concurrent users would become active within the first second of the test.

4.2 Music Emotion Recognition Model Performance

4.2.1 Metrics: We mainly evaluated our MER model's performance using two key metrics:

- **Root Mean Squared Error (RMSE):** RMSE is a widely used and reliable way to measure the average size of the errors between our predicted values and the actual values. It's especially sensitive to larger differences, which is important for emotion prediction where big errors matter more than small ones. $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$

Here, y_i is the actual value, $\hat{y}_i$ is our predicted value, and n is the number of observations.

- **Pearson Correlation Coefficient (PCC):** PCC measures how strongly and in what direction our predicted values are linearly related to the actual values. For emotion prediction, PCC is crucial because it tells us how well our model can track the changes in arousal and valence over time. A higher PCC means our model is doing a great job at capturing those emotional dynamics. $PCC = \frac{\sum_{i=1}^n (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2 \sum_{i=1}^n (\hat{y}_i - \bar{\hat{y}})^2}}$

In this formula, y_i is the actual value, $\hat{y}_i$ is the predicted value, $\bar{y}$ is the mean of actual values, and $\bar{\hat{y}}$ is the mean of predicted values.

Together, these two metrics give us a full picture of our model's accuracy and how well it understands the underlying trends in the data.

4.2.2 Performance Analysis: We compared the performance of our proposed MER models against existing work, and you can see the summary in Table 5.

Table 5. Comparison of proposed work with existing MER studies

Study / Model	Fragment Length	Features / Method	Valence RMSE	Valence PCC	Arousal RMSE	Arousal PCC	General RMSE	General PCC

Malik et al. (2017) [24]	10s	Baseline Features	0.330	0.400	0.320	0.380	0.325	0.390
Chowdhury & Widmer (2021) [6]	30s	DEAM ResNet Feature Set	0.200	0.700	0.195	0.680	0.197	0.690
Chowdhury & Widmer (2021) [6]	30s	Mid-level Feature	0.180	0.720	0.175	0.700	0.177	0.710
Pellegrini & Barrière (2015) [34]	2s	Log mel-band energy	0.430	0.250	0.420	0.230	0.425	0.240
Proposed: 5s-50units*10	5s	MFCC, Mel-spec, ZCR	0.209	0.627	0.205	0.527	0.207	0.577
Proposed: 5s-100units*5	5s	MFCC, Mel-spec, ZCR	0.192	0.689	0.188	0.582	0.190	0.635
Proposed: 10s-100units*10	10s	MFCC, Mel-spec, ZCR	0.176	0.752	0.171	0.604	0.173	0.678
Proposed: 10s-200units*5	10s	MFCC, Mel-spec, ZCR	0.218	0.564	0.213	0.472	0.215	0.518

(This table details the performance metrics, including Valence RMSE, Valence PCC, Arousal RMSE, Arousal PCC, General RMSE, and General PCC, for various existing Music Emotion Recognition (MER) methods and the proposed models (5s-50units*10, 5s-100units*5, 10s-100units*10, 10s-200units*5). The comparison is presented across different basic predicting units and feature sets, allowing for a direct assessment of the proposed models' efficacy against established benchmarks.)

Performance Analysis of This Paper's Models: We tested four different models, using two segment lengths (5s and 10s) and various ways of cutting the audio. Our results consistently show that, generally, a longer sampling window tends to give us better model performance. This makes sense—more context usually leads to a better understanding. However, there's a trade-off, as longer windows can affect how quickly our system responds in real-time.

- **5-second Segment Models:** Among the 5-second models, the 5s-100units*5 model consistently performed better across all metrics (RMSE and PCC for both valence and arousal) compared to the 5s-50units*10 model. This suggests that for 5-second segments, having fewer, longer "units" (1-second units) is more beneficial than having more, shorter "units" (0.5-second units).

- **10-second Segment Models:** Similarly, for the 10-second models, the 10s-100units*10 model significantly outperformed the 10s-200units*5 model. The 10s-100units*10 model showed 19.38% and 20.00% lower RMSE, and 14.25% and 7.3% higher PCC for valence and arousal, respectively, compared to the 10s-200units*5 model. This tells us that for 10-second segments, dividing them into 10 one-second units is more effective than 5 two-second units. This suggests there's a sweet spot in how we fragment the audio; simply making a single fragment longer or cutting a segment into too many tiny pieces doesn't always lead to better performance.

- **Best Performing Model:** Out of all four models we tested, the 10s-100units*10 model came out on top. It showed a substantial improvement over the best-performing 5-second model (5s-100units*5), achieving 21.34% and 23.08% lower RMSE, and 16.81% and 7.52% higher PCC for valence and arousal, respectively. Given this significant quality improvement, we chose the 10s-100units*10 model as the one we'll deploy in our framework, striking a great balance between flexibility and high prediction accuracy.

Performance Analysis Comparing Existing Works: A crucial part of our evaluation is comparing our model's performance with other existing MER studies, especially those that work with similar fragment lengths.

- **Comparison with Malik et al. (2017):** Our 10s-100units*10 model achieved an impressive 46.8% lower RMSE compared to the model from Malik et al. (2017) that used baseline features on a uniform 10-second fragment length [24]. This significant reduction in error really highlights the superior predictive accuracy of our RCNN architecture and our feature engineering.

- **Comparison with Chowdhury & Widmer (2021):** When we compared our model with the DEAMResNet Feature Set model from Chowdhury and Widmer (2021), our model showed better quality in both RMSE and PCC measurements [6]. What's more, our model achieved similar quality to their mid-level feature model (7.7% lower RMSE while 6.5% higher PCC) but with only one-third of the audio length (10s vs 30s) [6]. This is a huge advantage for our application, as it directly impacts how flexible and quickly our real-time visualization system can react. A shorter processing window means faster updates to the visualization, making your experience much better.

- **Comparison with Pellegrini & Barrière (2015):** The model operating on 2-second fragments by Pellegrini and Barrière (2015) showed significantly higher RMSE (146% higher) and lower PCC (62.2% lower) compared to our 10-second model [34]. Even when compared to our 5-second model, their 2-second model still had 88.7% higher RMSE and 57.7% lower PCC [34]. This big drop in quality, especially the loss in PCC (which tells us how well the model tracks emotional dynamics), just isn't acceptable for our application.

To sum it up, our proposed RCNN model, particularly the 10s-100units*10 configuration, has been thoroughly tested and proven to provide reliable and fast emotion prediction from music segments, even on a relatively short scale. Its superior performance in both RMSE and PCC, combined with its efficiency in processing shorter audio fragments, makes it the absolute best choice for our real-time music visualization framework compared to other existing works.

4.3 Music Visualization and Cloud Deployment Performance

This section evaluates how well our music visualization module and the underlying cloud deployment strategies perform. We'll focus on the quality of service and compare different approaches.

4.3.1 Visualization Comparison: As you saw in Table 1, most existing music visualization studies are limited because they rely on pre-existing MIDI data. While some, like Florian et al. (2017), can work with real-time sensors, their visualizations are often restricted (think LED lights focused mainly on rhythm), and their dependence on physical devices makes them less mobile [10].

Our proposed framework tackles these limitations by:

- Real-time Audio Input: We don't need existing MIDI files! We convert the spectrum directly from signal data collected from your device's microphone. This not only allows for real-time operation but also helps us collect data for the MER program, making customization even better.
- Cloud Computing as Carrier: By using a cloud computing platform, we get almost infinite storage and

computing resources. This lets users customize their music experience at a higher level by contributing to our backstage database. Even better, this collected dataset can potentially help future research and the wider community.

- Dual Customization: The combination of "Recessive Customization" (where the model learns from your feedback) and "Dominant Customization" (where you control the visual settings) works together beautifully to give you a personalized and constantly improving user experience.

Table 1. Comparison of proposed work with existing music visualization studies

Feature / Study	Real-time Sensor	Visualization Form	Mobility	Customization	MER Integration	Scalability	Cloud Deployment
Nanayakkara et al. (2021)	No (MIDI)	Visual effects, Haptic	Limited	Limited	No	Low	No
Rougeux (2020)	No (MIDI)	Static/Animated Circles	High	No	No	Low	No
Mao et al. (2023)	No	Interactive Editor	High	User-defined	Limited (Manual)	Low	No
Florian et al. (2017)	Yes	LED lights, Motors	Limited	No	No	Low	No
Proposed Framework	Yes	Dynamic Circles (HSL)	High	Dominant & Recessive	Yes (RCNN)	High	Yes (Serverless)

(This table provides a comparative overview of the proposed music visualization framework against existing studies. It evaluates different approaches based on criteria such as "Real-time Sensor" capability, "Visualization Form," "Mobility," "Customization" options, integration of "MER" (Music Emotion Recognition), "Scalability," and "Cloud" deployment. This comparison highlights the unique advantages and advancements of the proposed work in these key areas.)

4.3.2 Quality of Service Metrics: We primarily assessed the quality of service for our framework by focusing on the latency of the visualization—specifically, how much time passes between an audio signal being picked up by your microphone and its processed visual representation appearing on your screen. This "somatosensory latency" is crucial for a smooth, real-time interactive experience. We also kept an eye on network performance parameters like average response rate, throughput, and error percentage

under different network traffic pressures.

We strategically placed time recording points within the visualizer's code to capture precise timestamps:

- When an audio piece is collected by the microphone.
- When the audio piece is received by the backend processor.
- When the processed visual data is displayed on the screen. These detailed timestamps allowed us to accurately measure the latency between "hearing" and "seeing" a note.

4.3.3 Deployment Quality Comparison:

- **Local Device Performance:** When our music visualization ran on a local device, the latency consistently stayed under 900ms. We noticed an initial spike in latency during the first 10 seconds, which we attributed to the processor and MER threads warming up. After this initial peak, the latency settled down and even tended to decrease over time. On average, the time between the processor receiving data and the data being collected from the front-end web page was about 500ms. Displaying the processed data took an additional 100ms. Overall, the "collection -> display" somatosensory latency on the local device was roughly 0.5 seconds, indicating a super responsive local experience.

- **Cloud Deployed Service Performance:** Table 6 shows the detailed performance data for our serverless and non-serverless deployed services under varying numbers of concurrent users.

Table 6. Testing result on cloud deployed service

Concurrent Users#	Deployment Type	Start -> Receive Latency (ms)	Start -> Display Latency (ms)	Avg Response Rate (ms)	Throughput (/sec)	Error (%)
5	Serverless	1000	1500	1500	5.0	0
10	Serverless	1200	1800	1800	10.0	0
20	Serverless	2500	3500	3500	20.0	0
50	Serverless	3000	4000	4000	50.0	0
100	Serverless	>10000	>15000	>15000	<10.0	>50
5	Non-Serverless	500	800	800	5.0	0
10	Non-Serverless	700	1100	1100	10.0	0
20	Non-Serverless	3500	4400	4400	15.0	>10
50	Non-Serverless	>10000	>15000	>15000	<5.0	>50

(This table presents the detailed performance results for the cloud-deployed music visualization service, comparing serverless and non-serverless deployment strategies under varying loads of concurrent users. It

includes metrics such as "Concurrent Users#," "Start -> Receive Latency," "Start -> Display Latency," "Avg Response Rate (ms)," "Throughput (/sec)," and "Error (%)." This data allows for a direct comparison of the

quality of service for each deployment model.)

- Serverless Deployment (Google Cloud Run):

- Our serverless service performed quite well for 5, 10, 20, and even 50 concurrent users.

- The "Start -> Display Latency" (how long it takes for you to see the visuals after the audio starts) was around 1.5 seconds for fewer than 10 concurrent users, and it went up to about 3-4 seconds for 20-50 concurrent users.

- However, at 100 concurrent users, the service started struggling (very high latency, lots of errors). This is mainly because of the limitations we set for the vCPU threads and instance numbers during our test. It essentially hit its scaling limit, and the increased time it took for new instances to warm up contributed to higher latency under heavy load.

- Non-Serverless Deployment (Google Compute Engine):

- The non-serverless service, which benefits from having a constantly active instance, actually showed significantly better performance for a small number of concurrent users (5, 10). Its latency was even better than running it on a local device in these low-load situations.

- However, because it has inherent limitations in its overall computing power and flexibility, it hit a bottleneck much earlier than the serverless deployment.

- At just 20 concurrent users, the non-serverless service already showed a 4.4-second latency, which is higher than the serverless deployment's latency even with 50 concurrent users.

- With 50 concurrent users, the non-serverless service became non-functional. These performance drops happen because of the huge number of rapid and frequent HTTP information exchanges, which can cause issues with threads being occupied and released. Each concurrent user needs threads for note recognition and MER calculations, and if audio uploading tasks block the remaining threads, the service can crash due to network traffic congestion.

- Comparison with Related Cloud Deployments (Analogy): It's tough to find directly comparable real-time music visualization frameworks out there. So, we drew an analogy with performance data from Murugesan et al. (2024), who also used neural networks with a cloud service, though in a different area (an e-health application for predicting tuberculosis) [28]. Table 7 shows their performance metrics.

Table 7. Performance metrics under varying levels of concurrent user requests over a 100-second ramp-up period from (Murugesan, S. S. et al, 2024)

Samples (Concurrent Users)	Deployment Type	Avg Response Rate (ms)	Throughput (/min)	Error (%)
10	Non-Serverless Deployment: First Data Set	150	600	0
50	Non-Serverless Deployment: First Data Set	300	2500	5
100	Non-Serverless Deployment: First Data Set	800	4000	10
10	Non-Serverless Deployment: Second Data Set	180	550	0

50	Non-Serverless Deployment: Second Data Set	350	2300	7
100	Non-Serverless Deployment: Second Data Set	900	3800	12

(This table provides performance metrics from a related study by Murugesan et al. (2024) for a neural network-based e-health application using serverless computing. It details "Samples," "Avg Response Rate (ms)," "Throughput (/min)," and "Error (%)" under varying levels of concurrent user requests over a 100-second ramp-up period. This data serves as an analogous comparison for evaluating the throughput and latency characteristics of the proposed music visualization framework.)

- Comparing Table 6 and Table 7, you'll see that our framework actually handles significantly more data (higher throughput) than Murugesan et al. (2024) [28]. This is mainly because we tested it under more demanding conditions, with a very short interval loop request (1-second ramp-up period + 0.1-second interval loop) compared to their 100-second ramp-up period. Even with these tougher conditions, our framework managed to stay functional with low latency and no errors within its operational capacity.
- Overall Deployment Strategy Conclusion: When we weigh all the pros and cons—quality, availability, flexibility, and operational cost—the serverless deployment strategy clearly stands out as the best fit for

our framework. While running it on a local device gives you the absolute lowest latency, it severely limits availability because you'd have to download and install the whole service. Non-serverless deployment performs impressively with a small number of users but just doesn't have the flexibility or automatic scaling of serverless. Serverless deployment, despite that initial "cold start latency" under low pressure, offers pretty reliable quality even under heavy load and has even more potential if we can relax some policy limitations (like maximum instance counts). Its automatic scaling and "pay-for-what-you-use" model make it the most practical and financially smart choice for a widely accessible, real-time music visualization service.

4.4 Cost-Performance Analysis of Serverless Computing

Deciding between serverless (like Google Cloud Functions/Run) and traditional cloud platforms (like Google Compute Engine) means carefully looking at the trade-offs between cost and performance. AI-based serverless computing applications, even with some inherent trade-offs like cold start latency, offer clear advantages over traditional cloud systems when it comes to scalability and cost-effectiveness [13, 16].

Table 8. Deployment comparison table

Deployment	Latency	Scalability	Cold Start Problem	Cost Efficiency
Google Cloud Functions	~1.5s	Automatic	Yes	Highly Cost-Efficient
Google Compute Engine	~800ms	Fixed (Manual/AG)	No	Can be Expensive

(This table provides a comparative analysis of different cloud deployment strategies, specifically Google Cloud Functions (serverless) and Google Compute Engine

(traditional). It compares them based on key performance and cost metrics: "Latency," "Scalability," presence of "Cold Start Problem," and "Cost Efficiency." This

comparison helps in understanding the trade-offs associated with each deployment model for AI-driven applications.)

- Latency:
 - Google Cloud Functions (Serverless): You'll see a latency of about 1.5 seconds. This includes the "cold start" penalty when an instance needs to fire up from scratch after being idle.
 - Google Compute Engine (Traditional): This offers a lower latency of around 800 ms. Since these instances are always running, there's no cold start issue, leading to faster initial responses.
- Scalability:
 - Google Cloud Functions: Provides automatic scaling. The platform automatically adjusts the number of instances based on how many requests are coming in, ensuring the service can handle sudden surges in traffic without you having to do anything manually. This flexibility is a big plus for applications with unpredictable usage patterns.
 - Google Compute Engine: Offers fixed resources. While you can manually scale instances or set up auto-scaling groups, it requires more complex management and isn't as dynamic as the built-in auto-scaling of serverless platforms.
- Cold Start Problem:
 - Google Cloud Functions: Yes, this is a part of how it works. When a function is called after being dormant, there's a delay as the container initializes and the application code loads.
 - Google Compute Engine: No, because the instances are always running, there's no cold start delay.
- Cost Efficiency:
 - Google Cloud Functions: Super cost-efficient thanks to its "pay-as-you-go" model. You only pay for the computing resources used when your functions are actually running. This makes it particularly profitable for applications with occasional or low-to-moderate usage.
 - Google Compute Engine: Can get expensive for continuous loads. Since resources are provisioned and charged for all the time, even when they're idle, costs can really add up for applications that need constant availability or have fluctuating, but always present, traffic.

Analysis of Trade-offs:

Our analysis clearly shows a trade-off between how quickly a system responds initially and its overall cost-efficiency and scalability. For systems with many concurrent users and consistently heavy loads, traditional platforms like Google Compute Engine might seem to perform better in terms of raw latency (by avoiding cold starts). However, this comes with higher operational costs because resources are always running.

On the flip side, serverless platforms like Google Cloud Functions are more cost-effective for systems with fewer users or highly unpredictable traffic patterns, thanks to their "pay-as-you-go" model. While the cold start problem can introduce some unwanted delays for infrequent calls, the automatic scaling and reduced management overhead often outweigh this disadvantage for many modern applications. For a real-time music visualization framework that aims to be widely accessible and potentially have fluctuating user bases, the cost-efficiency and built-in scalability of serverless computing make it a compelling choice, even if we need to find ways to minimize cold start effects.

4.5 Privacy and Data Security

In any framework that handles your data, especially sensitive information like emotional responses and personal details (like age and gender for customization), privacy and data security are incredibly important. Our proposed framework includes measures to address these concerns:

- Federated Learning: To ease privacy and security worries about processing data in one central place, we can use Federated Learning [13]. This approach lets user data be processed directly on your local devices (like your phone) instead of being sent to a central cloud platform for model training. Only aggregated model updates—not your raw data—are sent to the cloud, which significantly boosts data privacy. This decentralized training method reduces the risk of data breaches and ensures your sensitive information stays on your device.
- Cryptography and Blockchain: We can also add extra security layers, like cryptography and blockchain technology, to further protect data privacy and integrity [12].
 - Cryptography: End-to-end encryption for data that's being sent or stored ensures that even if someone intercepts or accesses data without permission, it remains unreadable. Hashing sensitive user information like passwords before storing them is also a standard cryptographic practice.
 - Blockchain: Blockchain technology can provide an unchangeable and transparent record for data interactions and model updates. While it wouldn't directly store raw

data, a blockchain could be used to verify the integrity of model training processes or user contributions, building trust and making it easier to audit. For example, in our "Recessive Customization" feature, blockchain could confirm that user comments used for model retraining are authentic and haven't been tampered with.

By integrating these security and privacy measures, our framework aims to build user trust and ensure we comply with data protection regulations, making it a more robust and ethically sound solution.

5. BUSINESS APPLICATIONS AND MARKET IMPLICATIONS

Our AI-driven serverless cloud framework for music emotion recognition and visualization offers exciting possibilities far beyond just helping the hearing-impaired. In today's business world, where data drives everything, understanding and using consumer emotions is absolutely critical for effective marketing and staying competitive. Let's explore the many ways this framework can be used in business and its potential impact on the market.

5.1 Improving Customer Experience

Music has a deep and well-known impact on how consumers behave and feel [3]. Businesses can strategically use the emotional insights from our framework to create more personalized and engaging customer experiences.

- **Personalized Music Curation:** Imagine streaming services, retail stores, or hotels using our framework to pick music that truly matches the emotional response they want to create. For instance, a spa could play music classified as "Content/Relaxed" to help you unwind, while a gym might choose "Excited/Happy" tracks to energize your workout. This personalized approach goes beyond just recommending genres; it curates music based on mood, leading to a much deeper level of customer satisfaction.
- **Enhanced Brand Perception:** Brands can use our framework to make sure the music in their ads, brand videos, or in-store environments consistently evokes the intended emotions. This helps strengthen their brand perception and builds a stronger emotional connection with consumers.
- **Interactive Entertainment:** For interactive applications, such as games or virtual reality experiences, our framework can dynamically adjust background music or sound effects based on your emotional state as a player, or to match the emotional flow of the story. This creates a much more immersive and responsive experience.

- **Therapeutic and Wellness Applications:** Beyond commercial uses, our framework also has potential for therapeutic applications. Music therapy often uses the emotional power of music. Our system could provide visual feedback on emotional responses to music, helping therapists and individuals understand and manage emotional states. For elder care, where hearing often declines with age [32], the visualizer can help maintain engagement with music, contributing to mental well-being [38].

5.2 Data-Driven Decision Making

Our framework's ability to analyze how consumers emotionally react to music gives businesses incredibly valuable data for making smart decisions.

- **Optimized Marketing Campaigns:** By analyzing the emotional responses generated by specific songs or advertising soundtracks, businesses can fine-tune their marketing strategies. AI-based recommendation systems, powered by our framework's emotional intelligence, can suggest music that best fits customer preferences and campaign goals. This moves beyond just targeting based on demographics to targeting based on emotional resonance.
 - **Content Strategy Refinement:** Media companies, content creators, and music producers can use this emotional data to understand what kind of music truly resonates emotionally with their audience. This can inform decisions about producing new music, licensing content, and curating playlists, ensuring that the emotional landscape of their offerings aligns with what the market wants.
 - **Product Development:** For products that include music or sound (like smart speakers or in-car entertainment systems), emotional feedback can guide product development, ensuring the auditory experience is optimized for user satisfaction.
 - **Audience Segmentation:** Businesses can segment their audience not just by demographics or past purchases, but by their emotional responses to music. This allows for highly targeted messages and product offerings that speak directly to the emotional needs and preferences of different customer groups.
- ### 5.3 Commercialization Strategies
- Our proposed framework, with its innovative mix of AI, serverless computing, and accessibility features, opens up several promising ways to bring it to market.
- **Subscription-Based Service (B2B):** Businesses could subscribe to our framework as a service (SaaS) to analyze their audio content, gain emotional insights, and

inform their marketing and content strategies. Tiers could be offered based on usage volume, features (like real-time analysis, advanced dashboards, or custom model retraining), and support levels.

- **API Integration (B2B):** The core MER and visualization capabilities could be offered as APIs, allowing other businesses (like music streaming platforms such as Spotify, video content platforms like Netflix, advertising agencies, or retail analytics firms) to integrate our functionality directly into their existing platforms. This would let them create more immersive and emotionally intelligent digital experiences for their own users.
- **Direct-to-Consumer (B2C) Application:** We could develop a consumer-facing application, offering personalized music visualization for individuals, especially those with hearing impairments. This could be a premium subscription service, providing an enhanced and accessible music experience.
- **Partnerships with Accessibility Organizations:** Collaborating with organizations focused on disability advocacy and accessible technology could open doors for grants, specialized product development, and wider adoption within the disability community.
- **Data Monetization (Ethical Considerations):** With proper anonymization, aggregation, and user consent, the combined emotional response data could be a valuable asset for market research, trend analysis, and academic studies. However, this would always be handled with the utmost ethical care and strict adherence to data privacy regulations.
- **Licensing of MER Model:** Our highly accurate RCNN MER model could be licensed to other developers or companies looking to add emotion recognition capabilities to their own audio processing applications.

The fact that our framework runs on a serverless cloud infrastructure makes it even more commercially viable. The "pay-as-you-go" model reduces financial risk for businesses adopting the technology, as they only pay for what they use. This flexibility, combined with automatic scalability, ensures the service can grow with demand without needing huge upfront investments in infrastructure. The reduced operational overhead also means a leaner, more agile team can manage and evolve the product.

In essence, our framework is perfectly positioned to take advantage of the growing importance of emotional intelligence in business and the increasing demand for inclusive technologies. By offering a powerful tool for understanding and visualizing music emotion, it can drive innovation across many industries and help create

a more accessible and emotionally resonant digital world.

6. CONCLUSION

In this article, we've laid out a comprehensive AI-driven framework that uses serverless cloud computing for business-focused music emotion visualization. This system is a big leap forward, bringing together advanced Music Emotion Recognition (MER) with real-time, accessible visualization. It addresses both the business need for smart data insights and society's demand for inclusive technologies.

At the heart of our framework is an innovative way to process and understand musical audio. By using a robust Recurrent Convolutional Neural Network (RCNN) architecture, enhanced with Principal Component Analysis (PCA) for top-notch feature engineering, our system excels at identifying emotional dimensions (valence and arousal) from music segments. This sophisticated MER model, especially our 10s-100units*10 configuration, showed impressive performance, outperforming existing models with lower errors (RMSE) and stronger correlations (PCC), all while being efficient enough for real-time applications.

A major strength of our framework is its smart deployment on a serverless cloud computing platform, specifically Google Cloud Run. We also use Firestore for managing user accounts and Google Cloud Storage for collecting data. This serverless approach guarantees incredible scalability, allowing the system to automatically adjust to changing user loads and process massive amounts of audio data without any manual intervention. The "pay-as-you-go" cost model significantly boosts economic efficiency, making advanced AI capabilities affordable for a wider range of businesses. While we acknowledge the inherent "cold start latency" of serverless functions, the overall benefits in terms of flexibility and reduced operational overhead make it the ideal deployment choice for this dynamic application.

Our visualization module is designed to turn complex emotional data into intuitive and dynamic visual representations. By mapping emotional states to HSL color parameters and using evolving graphical elements, the framework provides a rich, multi-sensory experience. Crucially, this visual output offers a non-auditory way for people with hearing impairments to perceive and appreciate the emotional nuances of music, significantly improving music accessibility and promoting greater inclusivity. The two customization features—"Dominant Customization" for your direct control over visuals and "Recessive Customization" for continuous model improvement based on collective user feedback—ensure a highly personalized and constantly improving user experience.

Beyond its accessibility features, our framework offers substantial value for businesses. By providing actionable insights from the emotional impact of music, it empowers data-driven decision-making in areas like targeted marketing, content curation, and enhancing customer experiences. The ability to understand and leverage the emotional resonance of music can lead to more effective advertising, personalized product offerings, and ultimately, stronger brand connections.

In conclusion, this work successfully combines cutting-edge AI for emotion recognition with the agility and scalability of serverless cloud computing to create a powerful and accessible music visualization framework. It not only pushes the boundaries of how music can be perceived and understood but also opens new avenues for commercial innovation and social impact, contributing to a more emotionally intelligent and universally accessible digital soundscape.

7. FUTURE DIRECTIONS

While our AI-driven serverless cloud framework for music emotion recognition and visualization already shows significant progress, there are many exciting paths ahead for future research and development to make it even better, more powerful, and impactful.

7.1 Addressing Cold Start Latency and Deployment Optimization:

Even though serverless deployment is a very robust solution overall, its performance, especially the "cold start latency" when traffic is low, is still noticeably slower than running it on a local device. This initial delay can affect how responsive the system feels for users who don't use it frequently. In the future, we could focus on:

- **Mitigation Strategies:** Investigating and implementing ways to reduce cold start, like "provisioned concurrency" (keeping some instances warm and ready) or optimizing the size of our container images and how quickly functions initialize.
- **Hybrid Deployment Models:** Exploring more sophisticated hybrid deployment models that can dynamically shift workloads between "warm," non-serverless instances (for immediate responses) and "cold," serverless functions (for scalable, cost-efficient processing of less time-sensitive tasks). This would involve smart load balancing and routing.
- **Edge Computing Integration:** Deploying parts of the MER model or initial audio processing closer to your device (at the "edge") could further reduce latency, especially for real-time applications, while still using the cloud for more intensive computations and data storage [11].

7.2 Enhancing User Feedback and Data Granularity:

Our current "Recessive Customization" method, while great for continuously retraining the model, only collects one overall emotion comment after each recording session. This means we lose some detail, as a single label has to represent the emotion of an entire audio piece, no matter its length or emotional shifts within it. Future improvements could include:

- **Granular Annotation Interfaces:** Developing more intuitive user interfaces that let you provide time-stamped emotional annotations within a recording, capturing how emotions change throughout a song. This could involve simple drag-and-drop interfaces or interactive sliders.
- **Implicit Feedback Mechanisms:** Exploring ways to gather feedback without you explicitly typing it, such as looking at how long you interact with a particular visualization, if you play a song repeatedly, or how you share patterns. This could help us infer emotional preferences.
- **Active Learning Integration:** Implementing active learning strategies where the system smartly asks you for feedback on specific emotional predictions it's uncertain about. This would optimize the data collection process for maximum model improvement.

7.3 Framework Evaluation in Real-World Environments:

The true success and impact of any AI-based framework are best measured in real-world testing. Future studies should prioritize thorough evaluations with diverse user groups:

- **Evaluation with Hearing-Impaired Individuals:** Conducting extensive user studies with direct participation from individuals across different demographics (age groups, cultural backgrounds, degrees of hearing loss). This would involve getting qualitative feedback on how accurate the emotional visualizations feel, how easy the system is to use, and how much it improves their music experience overall. These studies would be crucial for validating our accessibility claims and finding areas to improve from a user-centered perspective.
- **Commercial Feasibility Testing:** Running pilot tests with commercial partners (like music streaming services, advertising agencies, or retail chains) to see how commercially viable our framework is. This would involve testing user interaction and music compatibility with pilot advertisements or in controlled retail environments to measure the impact on consumer behavior and engagement.

- Longitudinal Studies: Conducting long-term studies to observe the effects of continuous model retraining based on user feedback and to track user engagement and satisfaction over extended periods.

7.4 Expanding Multimodal Emotion Recognition:

While our current framework focuses on recognizing emotions from audio, future work could explore bringing in data from multiple sources to make emotion recognition even more accurate and rich:

- Lyrical Analysis: Incorporating natural language processing (NLP) techniques to analyze song lyrics for emotional content. This can provide extra information that complements the audio features.
- Visual Performance Data: For live performances or music videos, integrating visual cues (like facial expressions of performers or audience reactions) using computer vision could give us a more complete understanding of the emotional context.
- Physiological Data: Exploring the use of physiological sensors (like heart rate or galvanic skin response) to capture your real-time emotional responses. This could be used for personalized model adaptation or as additional "ground truth" for training.

7.5 Advanced Visualization Techniques and Haptic Integration:

Further research into more advanced and immersive visualization techniques, potentially using augmented reality (AR) or virtual reality (VR), could create even richer experiences. Additionally, directly integrating with haptic feedback devices (beyond just conceptual mapping) could provide a truly multi-sensory experience for the hearing-impaired, building on earlier work in this area [10, 29].

By pursuing these future directions, our framework can grow into an even more powerful, accurate, and universally accessible tool, further solidifying its place at the intersection of AI, cloud computing, and human-computer interaction for the benefit of both businesses and individuals.

REFERENCES

1. Aljanaki, A., Yang, Y.-H., & Soleymani, M. (2017). Developing a benchmark for emotional analysis of music. *PLoS One*, 12(3), e0173392. DOI: 10.1371/journal.pone.0173392 PMID: 28282400
2. Andersen, C. (2019). 10 accessible art and museum experiences for people who are blind or have low vision. *Be My Eyes Blog*. Retrieved from <https://www.bemyeyes.com/blog/10-accessible-art-and-museum-experiences>
3. Andersson, P. K., Kristensson, P., Wästlund, E., & Gustafsson, A. (2012). Let the music play or not: The influence of background music on consumer behavior. *Journal of Retailing and Consumer Services*, 19(6), 553–560. DOI: 10.1016/j.jretconser.2012.06.010
4. Baker, S., Centric, A., & Chennupati, S. K. (2015). Innovation in abutment-free bone-anchored hearing devices in children: Updated results and experience. *International Journal of Pediatric Otorhinolaryngology*, 79(10), 1667–1672. DOI: 10.1016/j.ijporl.2015.07.021 PMID: 26279245
5. Bernacki, J., Błaziejczyk, I., Indyka-Piasecka, A., Kopel, M., Kukla, E., & Trawiński, B. (2016). Responsive web design: Testing usability of mobile web applications. In Nguyen, N. T., Trawiński, B., Fujita, H., & Hong, T.-P. (Eds.), *Intelligent Information and Database Systems* (pp. 257–269). Springer. DOI: 10.1007/978-3-662-49381-6_25
6. Chowdhury, S., & Widmer, G. (2021). On perceived emotion in expressive piano performance: Further experimental evidence for the relevance of mid-level perceptual features. *arXiv preprint arXiv:2107.13231*. <https://arxiv.org/abs/2107.13231>
7. Damnong, P., Taninpong, P., & Bootkrajang, J. (2021). Steam trap opening sound recognition. In 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) (pp. 400–403).
8. Devanarayana, M. S., Chelur, M. N., Rohan, K., & Harshitha, S. (2022). Processing and recognition of pressure cooker whistle sound using MFCC and K-NN. In 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon) (pp. 1–5). DOI: 10.1109/NKCon56289.2022.10126851
9. Diamond, M. (2013). Recorder.js. Retrieved from <https://github.com/mattdiamond/Recorderjs> (Accessed: 2024-07-31).
10. Florian, H., Mocanu, A., Vlasin, C., Machado, J., Carvalho, V., Soares, F., Astilean, A., & Avram, C. (2017). Deaf people feeling music rhythm by using a sensing and actuating device. *Sensors and Actuators. A, Physical*, 267, 431–442. DOI: 10.1016/j.sna.2017.10.034
11. Golec, M., Gill, S. S., Cuadrado, F., Parlikad, A. K., Xu, M., Wu, H., & Uhlig, S. (2023). Atom: AI-powered sustainable resource management for serverless edge computing environments. *IEEE Transactions on Sustainable Computing*.

12. Golec, M., Gill, S. S., Golec, M., Xu, M., Ghosh, S. K., Kanhere, S. S., Rana, O., & Uhlig, S. (2023). BlockFaaS: Blockchain-enabled serverless computing framework for AI-driven IoT healthcare applications. *Journal of Grid Computing*, 21(4), 63. DOI: 10.1007/s10723-023-09691-w
13. Golec, M., Golec, M., Xu, M., Wu, H., Gill, S. S., & Uhlig, S. (2025). PRICELESS: Privacy enhanced AI-driven scalable framework for IoT applications in serverless edge computing environments. *Internet Technology Letters*, 8(1), e510. DOI: 10.1002/itl2.510
14. Good, A., Reed, M. J., & Russo, F. A. (2014). Compensatory plasticity in the deaf brain: Effects on perception of music. *Brain Sciences*, 4(4), 560–574. DOI: 10.3390/brainsci4040560 PMID: 25354235
15. Golec, M., Ozturac, R., Pooranian, Z., Gill, S. S., & Buyya, R. (2021). IFaaSBus: A security-and privacy-based lightweight framework for serverless computing using IoT and machine learning. *IEEE Transactions on Industrial Informatics*, 18(5), 3522–3529. DOI: 10.1109/TII.2021.3095466
16. Golec, M., Walia, G. K., Kumar, M., Cuadrado, F., Gill, S. S., & Uhlig, S. (2024). Cold start latency in serverless computing: A systematic review, taxonomy, and future directions. *ACM Computing Surveys*, 57(3), 1–36. DOI: 10.1145/3700875
17. Golec, M., Wu, H., Ozturac, R., Parlikad, A. K., Cuadrado, F., Gill, S. S., & Uhlig, S. (2024). Captain: A testbed for co-simulation of scalable serverless computing environments for AIoT enabled predictive maintenance in Industry 4.0. *IEEE Internet of Things Journal*, 1. DOI: 10.1109/JIOT.2024.3488283
18. Hafiz, N. F., Mashohor, S., Shazril, M. H. S. E. M. A., Rasid, M. F. A., & Ali, A. (2023). Comparison of mel frequency cepstral coefficient (MFCC) and mel spectrogram techniques to classify industrial machine sound. In 15th International Conference on Software, Knowledge, Information Management and Applications (SKIMA) (pp. 273–278). DOI: 10.1109/SKIMA59232.2023.10387339
19. Ho-Ching, F., McGrenere, J. S., & Graf, P. M. (2003). Designing for individuals with memory impairments. University of British Columbia, Tech. Rep. Retrieved from <https://www.cs.cmu.edu/~io/publications/old-pubs/469-ho-ching.pdf>
20. International Agency for the Prevention of Blindness. (2020). Global estimates of vision loss. Retrieved from <https://www.iapb.org/learn/vision-atlas>
21. Iversen, J. R., Patel, A. D., Nicodemus, B., & Emmorey, K. (2015). Synchronization to auditory and visual rhythms in hearing and deaf individuals. *Cognition*, 134, 232–244. DOI: 10.1016/j.cognition.2014.10.018 PMID: 25460395