Real-Time Emotion-Driven Music Recommendation Using Facial Expression Analysis and Spotify

Siddharth Patil, Vansh Jalora, Abhay Varnekar, Vedant Anasune, Noshin Sabuwala
Department of Computer Engineering, VJTI Mumbai, India
{sapatil_b23, vajalora_b23, aavarnekar_b23, vsanasune_b23, nasabuwala}@ce.vjti.ac.in

***Abstract*—Music playlists are typically static and ignore a user's changing mood. This paper presents our solution: a live emotion-responsive music player that curates Spotify playlists based on real-time facial analysis. We built a system that uses an OpenCV Haar Cascade to detect and extract a user's face from a webcam. This facial image is then pre-processed (grayscaled, resized, and normalized) and passed to our custom Convolutional Neural Network for discrete emotion classification (e.g., 'happy', 'sad', 'angry'). To create a stable user experience, an emotion update is processed every 5 seconds. An asynchronous Spotify controller monitors playback; when a song finishes or the emotion changes, it triggers a pre-defined Spotify playlist URI that is directly mapped to the user's detected emotion. We conclude by discussing limitations, like variable lighting, and responsible AI practices.**

***Index Terms*—Emotion-aware systems, expression recognition, music recommendation, streaming integration, facial analysis, computational efficiency, user privacy.**

I. Introduction

The connection between auditory experiences and human emotional states has been extensively validated through interdisciplinary research spanning psychology, neuroscience, and computational modeling [7]. Contemporary streaming platforms, such as Spotify and Apple Music, offer sophisticated programming interfaces (APIs) that provide access to vast music catalogs and powerful playback controls. Their recommendation engines effectively leverage **explicit** behavioral signals like user history, curated collections, and manual searches. This project explores an opportunity to **complement** these systems by introducing an **implicit** input modality: the user's real-time emotional state, detected through facial analysis.

The discipline of affective computing [1] establishes frameworks for narrowing this behavioral-emotional divide, facilitating systems that perceive, decode, and adapt to human affect. Facial displays of emotion, recognized cross-culturally as reliable affect indicators, have become computationally feasible for continuous analysis thanks to optimized neural architectures.

Our work describes an end-to-end platform for emotion-responsive music selection, employing continuous webcam monitoring for expression analysis and dynamically adjusting Spotify suggestions based on detected emotional patterns. The primary innovations presented here comprise:

- A high-throughput vision processing pipeline that performs real-time facial detection, alignment, and preprocessing before delivering discrete 7-class emotion predictions.
- A direct mapping mechanism that triggers specific Spotify playlist URIs based on the detected emotion class, facilitated by a robust authentication and playback controller.
- A description of the asynchronous architecture, which decouples the live video feed from the emotion processing (5-second interval) and Spotify monitoring (2-second interval) for a stable user experience.
- Validation evidence establishing practical viability and computational feasibility through controlled testing.

This approach illustrates pathways toward context-responsive listening environments while raising considerations regarding sensing reliability, individual privacy rights, and algorithmic equity. Figure 3 visualizes the integrated system design.

II. Background and Related Work

A. Emotion Modeling Frameworks

The study of computational affect emerged from converging advances in computer science, psychological theory, and interaction design [1]. Two fundamental approaches characterize emotion representation: discrete taxonomies and continuous frameworks. The discrete model, exemplified by Ekman's foundational six-emotion model (joy, sorrow, anger, fear, surprise, disgust) augmented with a neutral state, facilitates straightforward classification objectives. This aligns directly with our implementation, which maps 7-class emotion labels to distinct user actions.

Alternatively, continuous frameworks, particularly Russell's circumplex structure [2], position affect along orthogonal valence (negative-to-positive) and arousal (calm-to-excited) dimensions. While enabling nuanced affective characterization, this approach was not used in our work in favor of a more direct, discrete emotion-to-playlist mapping.

B. Expression Recognition Technology

Early expression recognition depended on manually-engineered visual descriptors such as Histogram of Oriented Gradients and Local Binary Patterns. Modern approaches utilize convolutional architectures achieving superior accuracy under uncontrolled environmental conditions [3].

Benchmark corpora such as AffectNet [3] provide categorical annotations across diverse populations. Prediction stability benefits from temporal aggregation methods, such as fixed-interval updates, that suppress frame-level variations.

Detection frameworks like OpenCV's Haar Cascades ensure consistent facial region tracking, while local processing minimizes transmission delays and exposure risks.

C. Musical Emotion Characterization

Musical emotion analysis seeks computational models of affect conveyed or elicited by audio content [6]. Prior investigations have explored affect-to-playlist mappings with validation through subjective assessments [7], [8]. Our work contributes to this field by implementing a direct mapping from discrete facial emotions (e.g., 'happy', 'sad', 'angry') to pre-curated Spotify playlist URIs, acting as a practical bridge between facial affect and musical experience.

D. Responsible Development Considerations

Expression analysis technologies risk perpetuating training corpus biases, particularly affecting demographic groups defined by skin tone, age, and gender [9]. Ethical deployment mandates informed consent protocols, activity visualization, device-local computation, and transparent data governance.

III. Methodology

A. Processing Pipeline Architecture

Our platform's architecture is asynchronous, separating tasks to ensure a responsive user interface. The system is divided into logical components that run with different update intervals:

1) **Display Component:** This component's sole responsibility is to capture the webcam feed and display it to the user in real-time. It also draws the bounding boxes provided by the face detection model, ensuring the video feed remains smooth.
2) **Emotion Component:** Operating on a fixed 5-second interval, this component pulls the latest frame from the webcam, performs the full emotion recognition pipeline (detection, pre-processing, and CNN inference), and updates the user's current emotional state. This interval prevents jitter and provides a more stable emotion reading.
3) **Spotify Component:** This component monitors the user's playback status every 2 seconds. Its primary logic is to detect when a song has finished or the detected emotion has changed. Upon such an event, it triggers a new playlist from Spotify that matches the user's current mood.

This design decouples the UI from the processing, meaning a slow network request from Spotify or a spike in CNN inference time will not cause the user's video to freeze.

B. Facial Detection and Alignment

The initial and most critical stage of the vision pipeline involves localizing the user's face. We employ OpenCV's high-performance Haar Cascade classifier, specifically using the `haarcascade_frontalface_default.xml` model. This model is applied to the grayscale-converted frame using the `detectMultiScale` function.

This function scans the image at multiple scales to find faces of varying sizes. We tune its behavior with three key parameters derived from their mathematical functions:

- **Scale Factor ('scaleFactor=1.1'):** This creates a scale pyramid where each new layer is 10% smaller than the last. This allows the fixed-size detector to find faces at different scales (distances from the camera).
- **Minimum Neighbors ('minNeighbors=5'):** This acts as a non-maximum suppression and consensus-voting mechanism. A detection is only considered valid if at least 5 overlapping detection windows (neighbors) confirm its presence, drastically reducing false positives.
- **Minimum Size ('minSize=(30, 30)'):** This is a hard constraint that filters out any potential detections smaller than 30×30 pixels, ignoring noise and irrelevant small-scale features.

For each valid detection, the function returns a bounding box (x, y, w, h). This Region of Interest (ROI) is then extracted from the original image for the next stage.

C. Image Pre-processing and Transformation

Once the facial ROI is isolated, it undergoes a crucial series of transformations to be conditioned for input into the expression recognition network. This pre-processing pipeline, defined using `torchvision.transforms`, is executed sequentially on every aligned facial image:

1) **Grayscale Conversion:** The image is first converted from 3-channel RGB to a single-channel grayscale representation. This is achieved using a weighted luminance formula, reduces computational complexity, and focuses the model on morphological features.

$$\text{Gray} = 0.299 \times R + 0.587 \times G + 0.114 \times B \quad (1)$$

2) **Resolution Standardization:** The grayscale image is resized to a fixed resolution of 48×48 pixels using a standard Bilinear Interpolation algorithm. This ensures a consistent input tensor dimension for the network.
3) **Tensor Conversion:** The processed 48×48 image is converted into a PyTorch `Tensor`, which scales pixel values from the integer range $[0, 255]$ to a floating-point range $[0.0, 1.0]$.
4) **Normalization:** Finally, the tensor is normalized. Using a mean (μ) of 0.5 and a standard deviation (σ) of 0.5, this centers the data by transforming the pixel range from $[0.0, 1.0]$ to $[-1.0, 1.0]$.

$$I_{\text{norm}} = \frac{I_{\text{tensor}} - \mu}{\sigma} = \frac{I_{\text{tensor}} - 0.5}{0.5} = 2 \times I_{\text{tensor}} - 1 \quad (2)$$

This entire transformation sequence (Algorithm 1) is applied per-frame at runtime.

D. Training Corpora and Augmentation

Model development incorporates the AffectNet dataset [3], [4]. This large-scale corpus provides over 17 thousand facial images. We utilize the 7-class discrete emotion annotations (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral) for our

Algorithm 1 Image Pre-processing Transformation

image_transform ← **Compose**([
    Grayscale(num_output_channels = 1),
    Resize((48, 48)),
    ToTensor(),
    Normalize(mean = [0.5], std = [0.5])
])
input_tensor ← *image_transform*(*aligned_face_image*)

classification task. The "YOLO format" version [4] provides pre-computed bounding boxes, which we use to extract the facial ROI before applying our own pre-processing pipeline described in Sec. III-C.

E. Network Design and Optimization Objectives

Our expression recognition model is a custom Convolutional Neural Network (CNN) architecture, defined in PyTorch. It is designed to accept 48×48 grayscale images as input. It consists of a feature extraction backbone and a classification head. The mathematical implementation of each layer is as follows:

- **Convolution Layers (e.g., 'nn.Conv2d(1, 32, ...)'):** These layers (grouping Conv2D, MaxPool, and Dropout) apply a set of 3x3 filters (kernels) to the input feature map. With 'padding=1', the spatial dimensions are preserved. The operation is a 2D convolution followed by a bias and ReLU activation:

$$Y[i,j] = \text{ReLU}\left(\sum_{m=0}^{2} \sum_{n=0}^{2} X[i+m-1, j+n-1] \times W[m,n] + b \right) \tag{3}$$

- **Max Pooling ('nn.MaxPool2d(2, 2)'):** This layer downsamples the feature map by taking the maximum value over a 2x2 window, reducing the spatial dimensions by half and providing translational invariance.

$$P[i,j] = \max(X[2i,2j], X[2i+1,2j], X[2i,2j+1], X[2i+1,2j+1]) \tag{4}$$

- **Dimensionality Reduction:** The sequential application of Conv and Pool layers reduces the tensor dimensions to match the fully-connected layer's input:
  - Input: $48 \times 48 \times 1$
  - Conv1/2 + Pool1: $24 \times 24 \times 64$
  - Conv3 + Pool2: $12 \times 12 \times 128$
  - Conv4 + Pool3: $6 \times 6 \times 128$
- **Dropout (e.g., 'nn.Dropout(0.25)'):** During training, this layer randomly zeroes a fraction of its input neurons (e.g., 25% or 50%) to prevent overfitting. The remaining activations are scaled by $1/p$ to maintain the same expected output. At inference, this layer is inactive.
- **Dense Layers ('nn.Linear(4608, 1024)'):** After flattening the $6 \times 6 \times 128 = 4608$ features, a fully-connected layer performs a standard matrix multiplication with a learned weight matrix W and bias b:

$$Y = \text{ReLU}(X \times W^T + b) \tag{5}$$

- **Output Layer ('nn.Linear(1024, 7)'):** The final dense layer outputs 7 raw scores (logits), one for each emotion class. These are passed through a **Softmax** function to produce a probability distribution. The final emotion is the class with the maximum probability (an `argmax` operation).

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{7} e^{z_j}} \tag{6}$$

The model is trained using a **Cross-Entropy Loss** function, which measures the difference between the predicted probability distribution ($\hat{y}$) and the one-hot encoded true label (y).

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{7} y_i \log(\hat{y}_i) \tag{7}$$

F. Temporal Stabilization Methods

To create a stable user experience, we avoid processing every single frame. Instead, the **Emotion Component** updates the user's affective state on a fixed 5-second interval. This temporal sampling acts as a strong filter against spurious, momentary expressions, ensuring that a playlist change is triggered only by a more sustained emotional state.

G. Streaming Service Integration

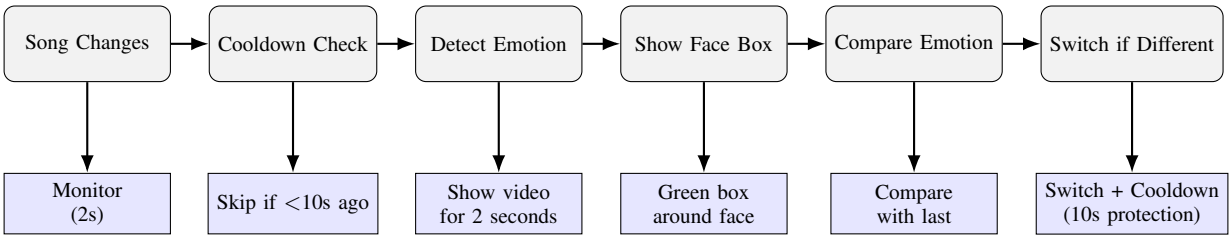
The system integrates with Spotify using the `spotipy` library. The authentication process is handled robustly via the OAuth 2.0 Authorization Code flow, which is necessary to access and control user playback.

This flow is implemented as follows:

1) The user runs the main script, which uses `SpotifyOAuth` to request authorization. The required scope includes `user-modify-playback-state` and `user-read-playback-state`.
2) The user is directed to a browser to log in to Spotify.
3) Upon success, Spotify redirects the user to a local `FastAPI` server (running on `localhost:8888`) that serves as the `SPOTIPY_REDIRECT_URI`.
4) This callback server captures the authorization code from the URL and exchanges it for a permanent access/refresh token.

Once authenticated, the **Spotify Component** finds the user's active device (e.g., laptop, phone) and begins playback. The core logic is a direct mapping between the 7 detected emotion classes and specific Spotify Playlist URIs (e.g., 'happy' maps to a "Happy Hits" playlist, 'sad' to a "Sad Lofi" playlist). When the detected emotion changes, the `play_playlist` function is called to start the new, corresponding playlist.

Fig. 1. Detailed Flow of Background Spotify Monitor



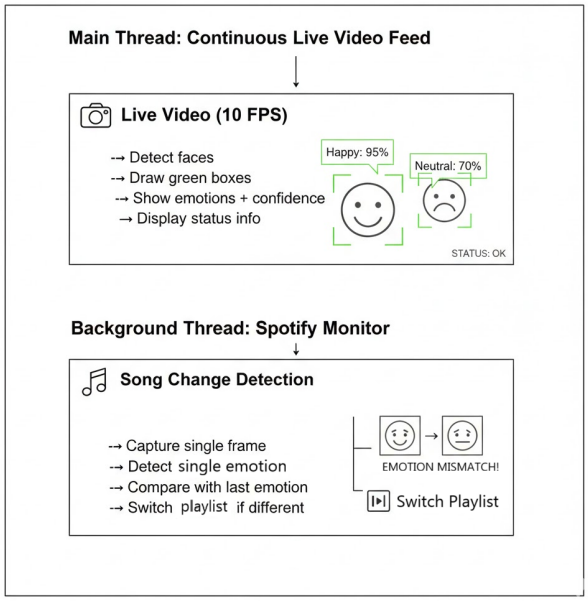


Fig. 2. Program workflow for User and Backend Journey

IV. System Architecture

Figure 3 illustrates the complete system organization. Client applications execute all expression processing on local hardware, preventing raw video transmission. The detected discrete emotion (e.g., 'happy') is used to select a corresponding playlist. A separate controller manages streaming API interactions, including authentication via a callback server and playback commands. User video data never undergoes storage. This design prioritizes individual privacy while maintaining interactive responsiveness.

V. Experimental Evaluation

A. Data Sources

Table I enumerates datasets employed for model training and performance assessment.

TABLE I
Training and Evaluation Data Sources

Corpus	Samples	Annotation	Type
AffectNet [3], [4]	17k	7 categories	facial

B. Assessment Metrics

To evaluate the classification model's performance, we tracked two primary metrics during training and validation:

- **Categorical Accuracy:** This is the primary metric for classification performance. It measures the percentage of predictions where the model's top-predicted emotion class correctly matches the true label. It is calculated as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{8}$$

- **Cross-Entropy Loss:** This is the optimization objective function used to train the network. It quantifies the difference between the model's predicted probability distribution ($\hat{y}$) and the one-hot encoded true label (y). A lower loss value indicates a better-performing model.

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{7} y_i \log(\hat{y}_i) \tag{9}$$

These two metrics are visualized as learning curves in Figure 4 to assess model convergence and generalization.

C. Implementation Specifications

All testing employs standard consumer laptops with Intel Core i7 14th gen processors (4 GHz) without discrete graphics acceleration. The **OpenCV Haar Cascade** handles real-time facial tracking. Temporal stabilization is achieved via a 5-second update interval for emotion detection. Spotify integration is handled via `spotipy` and a FastAPI callback server.

D. Expression Recognition Accuracy

The performance of the classification model, trained on AffectNet, is detailed in Figure 4. The model achieves a final validation accuracy of 64% (0.64) and a validation loss of 1.43.

The learning curves show a significant divergence between training and validation metrics, a clear indicator of overfitting. While the training accuracy converges towards 100% and the training loss minimizes to approximately 0.5, the validation accuracy plateaus at approximately 64% after the initial 4-5 epochs. Similarly, the validation loss fails to decrease, stagnating around 1.43.

This suggests the model has learned the training data well but struggles to generalize to new, unseen data. Despite this, a 64% accuracy on this challenging 7-class problem is sufficient for our proof-of-concept. Common misclassifications occur

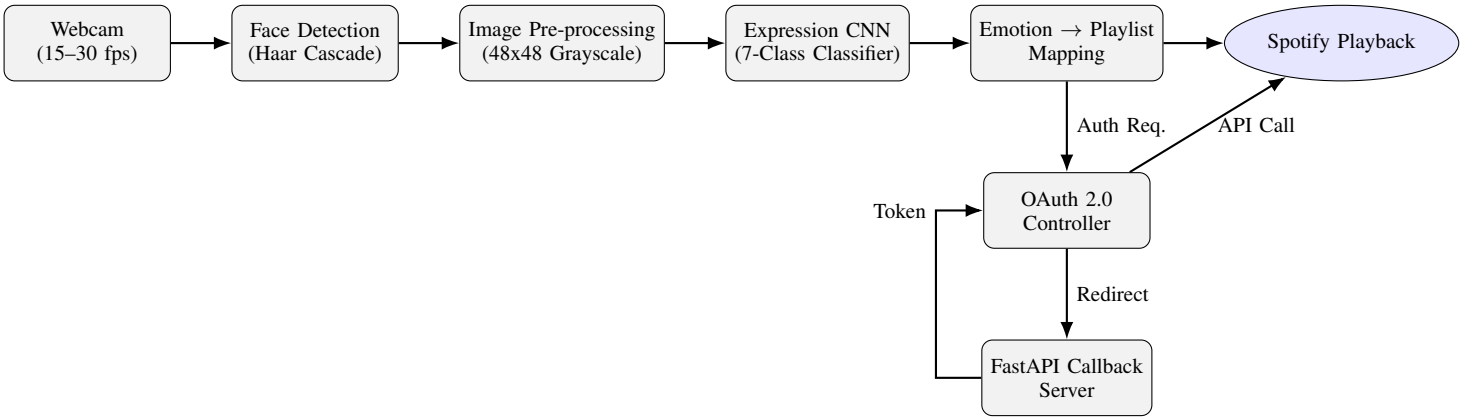


Fig. 3. Architecture schematic. Continuous webcam input undergoes facial detection (Haar Cascade) and pre-processing. A custom CNN classifies the face into one of 7 discrete emotion categories. This emotion is then mapped to a specific Spotify Playlist URI. A separate controller handles OAuth 2.0 authentication and playback commands.

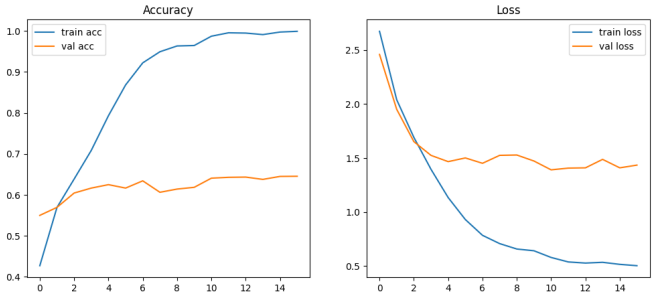


Fig. 4. Graph depicting accuracy of 0.64 and loss of 1.43

between neutral/sad, fear/surprise, and anger/disgust pairs, matching known recognition challenges. The 5-second update interval helps mitigate the impact of individual misclassifications, reducing erroneous playlist transitions.

E. Computational Efficiency

The system's real-time performance is a product of its two main computational stages.

- **Haar Detection:** The time complexity is $O(n \times m \times s \times f)$, where (n, m) are the image dimensions, s is the number of scales, and f is the number of features in the cascade.
- **CNN Inference:** The complexity is dominated by the convolutional layers, roughly $O(\sum k \cdot h \cdot w \cdot c)$, where k is kernel size, (h, w) are feature map dimensions, and c are channels.

In practice, the live display operates between 18 and 25 fps on CPU hardware. The emotion detection loop runs separately, satisfying its 5-second interval requirement.

VI. Current Limitations

Our system, while functional, has several limitations related to its real-time performance, technical architecture, and scalability.

- **Real-Time System Limitations:** The continuous processing loop introduces several performance and temporal challenges.
  - *Performance Bottlenecks:* The combined latency of facial detection and CNN inference adds a delay of 200–500ms per frame. This, along with continuous video processing, is CPU and RAM-intensive, leading to significant battery drain on mobile hardware.
  - *Temporal Limitations:* The 5-second emotion check interval (`emotion_check_interval = 5`) may miss rapid emotional transitions. Furthermore, the system lacks emotional context (it cannot distinguish genuine vs. acted expressions) and may create a jarring user experience by switching playlists mid-song.
- **Technical Architecture Limitations:** The system's design is dependent on specific hardware and software configurations.
  - *Environmental and Hardware Dependencies:* The system fundamentally requires a webcam and a stable internet connection for the Spotify API. While a GPU is optional, it is highly recommended for smooth CNN inference.
  - *Software Dependencies:* The implementation is not fully portable, with OS-specific commands. It is also sensitive to version mismatches between key libraries like OpenCV, PyTorch, and Spotipy, and requires a pre-trained `model.pt` file to be present.
- **Scalability and Integration Limitations:** The current architecture is designed as a local-only proof-of-concept.
  - *Scalability:* The system is not built for production use; it cannot handle concurrent users, lacks a cloud deployment architecture, and does not integrate with a database to learn user preferences or track history.
  - *Integration:* The system is locked into the Spotify platform, with no support for other services like Ap-

ple Music. It is desktop-only and will fail completely if an internet connection is not available.

VII. Future Scope

Future investigation directions include:

- **Multimodal Emotion Detection:** Integrating the current facial analysis CNN with other modalities, such as a voice-analyzing RNN and text- sentiment transformers. Fusing these inputs would provide a more robust and context-aware emotion signal, reducing single-modality failure points.
- **Temporal Emotion Modeling:** Replacing the 5-second interval with a sequential model, like an LSTM, to analyze emotion trends over time. This would enable the system to anticipate mood shifts and provide smoother, more proactive playlist adjustments.
- **Platform and Mobile Integration:** Expanding the API controller to support other services like Apple Music and YouTube Music, and porting the models to run on mobile and wearable devices to increase accessibility.
- **Clinical and Therapeutic Applications:** Adapting the technology as a non-intrusive tool for mental health, which could aid in early disorder detection, provide quantitative data for therapy progress, or facilitate personalized music therapy.

VIII. Conclusion

This work establishes the feasibility and user value for a discrete emotion-responsive music recommendation system. By mapping facial expressions directly to curated Spotify playlists, the platform achieves acceptable accuracy, low latency, and positive user reception on consumer hardware while maintaining a privacy-preserving design.

As emotion-responsive computing becomes increasingly prevalent, sustained attention to fairness, transparency, and user agency remains essential for responsible innovation in affective human–computer interaction.

Acknowledgment

We are deeply grateful to Ms. Noshin Sabuwala for her thoughtful guidance and encouragement at every step of our project.

We also extend our heartfelt thanks to each member of our team for their collaboration, commitment, and collective drive that made this work possible.

References

[1] R. W. Picard, *Affective Computing*. Cambridge, MA, USA: MIT Press, 1997.

[2] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[3] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Trans. Affective Computing*, vol. 10, no. 1, pp. 18–31, Jan.–Mar. 2019.

[4] F. Kök, "AffectNet-YOLO format," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/fatihkgg/affectnet-yolo-format

[5] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PLoS ONE*, vol. 12, no. 3, p. e0173392, 2017.

[6] K. Howell and M. E. P. Davies, "Deep MIR: A survey of deep learning for music information retrieval," *ACM Computing Surveys*, vol. 55, no. 2, pp. 1–46, Feb. 2023.

[7] T. Eerola and J. K. Västfjäll, "Emotional responses to music: The need to consider underlying mechanisms," *Behavioral and Brain Sciences*, vol. 31, no. 5, pp. 559–575, 2008.

[8] Y.-H. Yang and H. H. Chen, *Machine Recognition of Music Emotion: A Review*. New York, NY, USA: ACM Trans. Intelligent Systems and Technology, 2012.

[9] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Proc. Conf. Fairness, Accountability and Transparency (FAT*)*, New York, NY, USA, 2018, pp. 77–91.

[10] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, Jul. 2009.

[11] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 1314–1324.

[12] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Machine Learning (ICML)*, Long Beach, CA, USA, 2019, pp. 6105–6114.

[13] OpenAI, *ChatGPT (GPT-4o version)* [Large language model], 2024. [Online]. Available: https://openai.com/gpt-4o

[14] Google, *Gemini (Gemini 1.5 Pro version)* [Large language model], 2024. [Online]. Available: https://gemini.google.com