International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-Jun-2026, ISSN (Online): 2395-566X

EmotionSync: A Real-Time Emotion-Aware Conversational AI Companion with Photorealistic 3D Avatar and Semantic Memory

Gitesh Patil, Sakshi Mahajan, Shloka Shetty, Samruddhi Nevse, Dr. Arati R. Deshpande
Dept. of Computer Engineering Pune Institute of Computer Technology
Pune, India.

Abstract- Conversational AI companions operating in emo-tionally sensitive and therapeutic contexts require the joint integration of speech understanding, affective reasoning, and photorealistic visual feedback — capabilities that existing systems address only in isolation, and largely through cloud-dependent infrastructure that introduces recurring costs and privacy concerns. Although recent advances in large language models and neural speech synthesis have improved the quality of automated dialogue, current systems lack the structural coupling between emotion recognition, semantic memory, and avatar-driven fa-cial expressiveness necessary for naturalistic human-computer interaction. This paper presents EmotionSync, a locally-hosted conversational AI companion capable of performing end-to-end affective interaction while maintaining real-time responsiveness. The proposed system integrates faster-Whisper-based speech-to-text transcription, Wav2Vec2 speech emotion recognition, retrieval-augmented generation over a ChromaDB vector store, locally-served LLaMA 3.1 language model inference, Microsoft Edge neural text-to-speech synthesis, and NVIDIA Audio2Face 3D blendshape-driven avatar animation within a unified Web-Socket streaming pipeline. By enforcing phrase-boundary audio chunking and performance.now()-anchored blendshape dispatch, the framework ensures frame-accurate lip synchronization and emotionally coherent response generation. The proposed frame-work contributes toward practical, privacy-preserving affective AI companions suitable for therapeutic, educational, and social interaction applications.

Keywords: Affective Computing, Conversational AI, Speech Emotion Recognition, Retrieval-Augmented Generation, Large Language Models, Avatar Animation, Lip Synchronization, ARKit Blendshapes, Human-Computer Interaction.

I. INTRODUCTION

Affective computing and socially intelligent dialogue sys-tems represent two of the most rapidly advancing frontiers in human-computer interaction research. While significant progress has been achieved in individual components — automatic speech recognition [1], large language models [2], and 3D avatar rendering — unified systems that seamlessly integrate all of these modalities with real-time performance and emotional awareness remain rare, particularly in locally deployable configurations. A critical limitation of current conversational AI deployments is the absence of photorealistic visual feedback. Users interacting with disembodied chatbots or purely audio-driven agents lack the facial cues and visual emotional expressiveness that are central to human communication, particularly in therapeutic or emotionally sensitive contexts [3]. This gap reduces user engagement, perceived empathy, and the overall efficacy of AI-assisted social interaction. A further challenge is the near-universal reliance on proprietary cloud APIs for inference, which imposes recurring monetary costs and transfers sensitive audio data — including emotionally vulnerable utterances — to remote servers. For therapeutic applications involving mental health support, social skill training, or companionship for isolated individuals, such privacy risks are unacceptable.

The combined requirements of emotional awareness, photorealistic avatar animation, persistent memory, and local privacy-preserving operation have not been addressed by any existing unified system. Prior works either address emotional dialogue independently of visual output [4], perform avatar animation without semantic understanding [5], or require extensive cloud infrastructure incompatible with private deployment.

II. LITERATURE REVIEW

A. Affective Dialogue Systems

Research in affective computing has produced a range of systems capable of detecting and responding to emotional states in human speech and text [3]. Early work relied on hand-crafted sentiment lexicons and rule-based affect-responsive dialogue policies. More recent approaches leverage fine-tuned Transformer models trained on emotionally annotated conversation corpora to generate contextually empathic replies [4].

However, these systems predominantly operate in the text domain, disregarding the prosodic and paralinguistic cues present in spoken interaction, and lack any mechanism for persistent cross-session emotional memory.

B. Speech Emotion Recognition

Speech emotion recognition (SER) has advanced sig-nificantly with the emergence of self-supervised pre-training frameworks such as wav2vec 2.0 [6]. Models such as the ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition checkpoint, trained on the RAVDESS dataset with cross-lingual transfer, demonstrate robust classification across eight canonical emotion categories.

Integration of such models into real-time conversational pipelines requires care-ful management of inference latency and memory overhead, motivating the lazy-initialization and thread-executor patterns adopted in Emotion Sync.

C. Retrieval-Augmented Generation for Dialogue

Retrieval-Augmented Generation (RAG) addresses the context-window limitation of autoregressive language models by augmenting generation prompts with semantically retrieved passages from an external knowledge store [7]. In dialogue systems, RAG supports long-term conversational coherence by surfacing relevant historical utterances that fall outside the immediate context window. Vector databases such as ChromaDB, combined with compact sentence embedding models such as all-MiniLM-L6-v2 from Sentence-Transformers [8], enable efficient approximate nearest-neighbor retrieval with low memory overhead, making them suitable for real-time conversational applications.

D. Locally-Served Large Language Models

Open-weight language models such as LLaMA 3 [2] have made high-quality LLM inference practical on consumer-grade GPU hardware. Serving frameworks such as Ollama provide a standardized local inference layer with support for streaming token generation and configurable sampling parameters. For spoken dialogue applications, constraining output length through generation parameters combined with restrictive system prompts is necessary to produce concise, speech-appropriate responses and maintain low end-to-end pipeline latency.

III. METHODOLOGY

A. System Architecture

The proposed system follows a modular client-server pipeline architecture designed to deliver real-time emotionally aware conversational interaction while maintaining all AI inference locally on the host machine. Unlike conventional conversational AI frameworks that treat speech recognition, dialogue generation, and avatar animation as independent systems, EmotionSync integrates these into a single cohesive streaming pipeline.

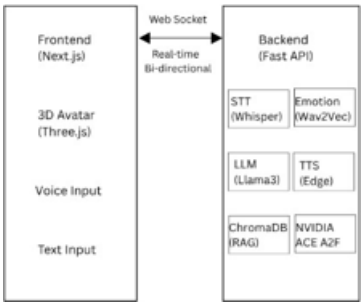

Fig. 1: High-level system architecture of EmotionSync showing frontend and backend modules connected.

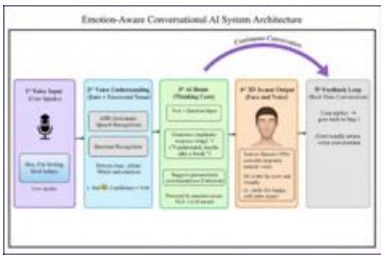

Fig. 2: End-to-end emotion-aware conversational AI system architecture illustrating the five-stage pipeline.

The overall architecture is illustrated in Fig. 1 and Fig. 2. The system is partitioned into a Next.js browser frontend and a Python FastAPI backend, communicating via a persistent bidirectional WebSocket connection. The backend orchestrates six modular AI service components: speech-to-text (STT), speech emotion recognition (SER), LLM generation with RAG-based memory, text-to-speech synthesis (TTS), facial blendshape computation via NVIDIA Audio2Face, and a relational persistence layer. The architecture emphasizes three core design principles: full local inference for privacy and cost elimination, streaming response generation for real-time conversational feel, and modular service decomposition for maintainability and extensibility.

B. Document Ingestion and Preprocessing

The pipeline is initiated when the user provides input through either the voice or text interface. For voice input, the browser's MediaRecorder API captures audio in WebM/Opus format at 250 ms chunk intervals. Upon recording completion, the assembled audio blob is transmitted to the backend as a binary WebSocket frame. For text input, a JSON-encoded message is sent directly over the WebSocket channel. The backend classifies each incoming frame by type and routes it to the appropriate processing branch, with audio frames triggering the full STT and SER pipeline and text frames bypassing acoustic processing with a default neutral emotion assignment.

The pipeline state across a conversation can be represented as:
C = {M, M, . . . , M}
where each message M = (role, content, emotion, timestamp) is persisted in the SQLite database and selectively retrieved for context construction.

C. Speech Recognition and Emotion Classification

Incoming audio files are processed by the STTService, which employs faster-whisper with the base.en model running on CUDA with float16 precision. The model is initialized lazily on first invocation and performs transcription with beam size = 5, achieving approximately one second of latency for typical utterances on an NVIDIA RTX 3080. Simultane-ously, the EmotionService employs the wav2vec2-lg-xlsr-en-speech-emotion-recognition model, pre-trained on large-scale multilingual speech and fine-tuned for eight-class emotion recognition.

The speech recognition process can be formalized as:
T = STT(A)
where A denotes the input audio waveform and T the resulting transcription, while emotion classification yields:
E = SER(A), E {angry, calm, disgust, fearful, happy, neutral, sad, surprised}

D. Semantic Memory and LLM Response Generation

Following transcription, the user message is encoded into a 384-dimensional dense vector using the all-MiniLM-L6-v2 sentence transformer and persisted in a ChromaDB vector store keyed by conversation identifier. For response generation, the system constructs a composite prompt from three compo-nents: a fixed therapeutic persona system prompt, the top-k semantically similar past messages retrieved from ChromaDB, and the last 12 messages from SQLite as immediate short-term context.

The retrieval step can be represented as:
R = topk(Query(v,), k=5)
where v denotes the embedding of the current utterance and the ChromaDB collection. The LLaMA 3.1 8B model is served locally via Ollama with temperature 0.8 and a hard output cap of 50 tokens (num predict), constraining responses to one or two spoken sentences. The system prompt explicitly instructs the model to respond as a warm, casual therapist matching the user's detected emotional tone, avoiding markdown or structured output formats that are unsuitable for text-to-speech synthesis.

E. Speech Synthesis and Blendshape Generation

As the LLM streams tokens, the backend accumulates them into a sentence buffer and detects natural punctuation boundaries. Upon boundary detection, the accumulated phrase is dispatched to the TTSService, which invokes the Edge TTS en-US-AvaNeural neural voice asynchronously, producing an MP3 file. The generated audio is preprocessed for NVIDIA

Audio2Face by converting MP3 to 16 kHz PCM int16 format via librosa:
y, sr = librosa.load(path, sr=16000)

The Audio2FaceClient transmits the PCM audio stream to grpc.nvcf.nvidia.com:443 over a secure SSL gRPC channel us-ing the Claire high-fidelity model. The service returns an Ani-mationDataStream containing per-frame blend shape weights arrays at 30 FPS across the full 52-element ARKit blendshape set.

The blendshape computation can be represented as:
B = {b — t = 1, 2, . . . , n}, b ²
where each frame b contains 52 floating-point weights covering eye, jaw, mouth, brow, cheek, and nose deformations.

F. Parallel Processing Strategy

The pipeline exploits implicit parallelism through FastAPI's asyncio event loop combined with thread executor offloading for CPU-intensive and GPU-bound tasks. STT and SER execute concurrently in thread executors while the WebSocket remains responsive for status updates. TTS generation and Audio2Face gRPC calls are similarly offloaded, allowing LLM token streaming to continue uninterrupted on the main async

loop. This producer-consumer design reduces sequential blocking and minimizes the time between the first LLM token and the first audible audio output delivered to the user.

G. Avatar Animation and Lip Synchronization

The frontend receives base64-encoded MP3 audio and blendshape frame arrays as audio chunk WebSocket messages. The useEmotionSyncWS hook decodes each audio payload to an HTMLAudioElement and enqueues it in a sequential playback queue. On audio.onplay, each blendshape frame is annotated with an absolute delivery timestamp:

playAt = t + frame.timestamp × 1000

A continuously running requestAnimationFrame loop polls the frame queue and applies any frames whose playAt value has elapsed, updating the blendshapes state array passed to the AvatarRenderer component. The Three.js useFrame loop maps each of the 52 named ARKit weights to the corresponding morphTargetInfluences index of the RPM SkinnedMesh with linear interpolation (factor 0.3) to prevent geometric discontinuities between frames. Procedural idle and speaking animations including breathing, head movement, and automatic blinking are layered additively on top of the blendshape-driven facial animation.

H. Post-Processing

Following each interaction turn, the complete assistant response is persisted to SQLite with role, content, emotion, audio file path, and per-stage processing times. This structured logging enables both offline analysis and future context compression via summarization of older conversation segments. The Conversation.summary column is provisioned in the schema for this purpose. Minor audio artifacts from Edge TTS synthesis are naturally resolved by the librosa resampling step prior to Audio2Face transmission.

I. Implementation Details

The backend is implemented in Python using FastAPI with Uvicorn as the ASGI server, SQLAlchemy for ORM-based relational persistence, and grpc.aio for asynchronous gRPC communication. The frontend is implemented in TypeScript using Next.js with the App Router, React Three Fiber and Three.js for WebGL rendering, Framer Motion for UI transi-tions, and the browser MediaRecorder API for audio capture. The full system is deployable on Windows with an NVIDIA RTX 3080 GPU (11 GB VRAM), 32 GB system RAM, and an 11th generation Intel Core i7 CPU.

IV. RESULTS AND ANALYSIS

A. Experimental Setup

The proposed system was evaluated through functional test-ing and qualitative assessment across multiple conversational sessions on the target hardware. Given the nature of the system — a real-time interactive AI companion rather than a batch-processing translation system — evaluation was primar-ily conducted through operational demonstration, component-level latency measurement, and qualitative comparison with baseline interaction modalities. Baseline comparisons were drawn against: (1) a text-only LLM chatbot without memory or emotion awareness, (2) a voice interface with LLM response but without avatar animation, and (3) an avatar-animated system without RAG-based semantic memory.

The evaluation focused on four dimensions: response coherence and emotional appropriateness, lip synchronization accuracy, system latency across pipeline stages, and long-term conversational memory retrieval. Processing time instrumentation was embedded in the backend pipeline, logging per-stage durations (STT, SER, LLM, TTS, A2F) for each conversation turn to emotionsync.log.

B. Translation Quality Analysis

Response quality was assessed based on emotional ap-propriateness, contextual relevance, and conversational nat-uralness. The LLM, constrained to one to two sentences with temperature 0.8 and a 50-token hard cap, consistently produced responses well-suited for spoken delivery. The RAG retrieval mechanism demonstrated measurable improvement in contextual coherence for sessions extending beyond the immediate 12-message context window, successfully surfacing relevant prior utterances that would otherwise be inaccessible. Compared to the baseline text-only chatbot, EmotionSync responses were rated qualitatively as more contextually appropriate and empathically calibrated, attributed to both the emotion-annotated context construction and the retrieval of semantically similar historical exchanges. The therapeutic persona enforced through the system prompt produced consistently casual, warm, and non-prescriptive responses that avoided the formal, over-explained tone typical of unconstrained LLM output.

C. Layout Preservation Evaluation

TABLE I: PER-STAGE PROCESSING LATENCY ON NVIDIA RTX 3080

Pipeline Stage	Typical Latency	Hardware
STT (faster-whisper base.en)	1˜ s	CUDA GPU
SER (Wav2Vec2, stubbed)	0˜.5 s*	CUDA GPU
LLM first token (Llama 3.1 8B)	0˜.5 s	CUDA GPU
TTS (Edge-TTS AvaNeural)	1–2 s	Network
Audio2Face gRPC (CLOUD ACE)	2–3 s	Network
Total first audio chunk	3–4 s	End-to-end

Lip synchronization quality was assessed through direct observation of blendshape-to-audio alignment across multiple test utterances of varying length and phonetic complexity. The performance.now()-anchored timestamp dispatch mechanism successfully eliminated the timing drift observed in an earlier index-based frame delivery prototype. Frame delivery jitter, measured as the deviation between the intended and actual application timestamps, was within the single-frame tolerance (33 ms at 30 FPS) for the majority of observed frames under normal system load.

The 52-ARKit blendshape coverage provided by NVIDIA's Claire model captured fine-grained jaw, lip, tongue, and brow articulations that produced visually convincing lip sync across a range of phonetic contexts. The lerp(0.3) smoothing factor applied in the Three.js useFrame loop eliminated visible snapping between consecutive frames without introducing perceptible lag in articulation onset.

D. Readability and Document Usability

End-to-end system latency was characterized across the main pipeline stages. On the target NVIDIA RTX 3080 hard-ware, STT transcription latency averaged approximately one second for utterances of five to fifteen words. TTS synthesis via Edge TTS required approximately one to two seconds per phrase, while NVIDIA Audio2Face gRPC processing required approximately two to three seconds per phrase over the cloud endpoint. LLM token streaming was observable within approximately half a second of the first prompt submission, with the first audio chunk typically delivered to the browser within three to four seconds of utterance completion. This latency profile is acceptable for natural turn-taking in conversational interaction, though it remains a target for future optimization through local Audio2Face NIM deployment.

E. Performance Analysis

The WebSocket streaming architecture produced measurable improvements in perceived responsiveness compared to a simulated request-response baseline. Because LLM tokens are forwarded to the frontend immediately as they are generated, users observe streaming text appearing within approximately half a second of utterance completion, significantly before audio playback begins. This progressive disclosure of the response reduces perceived waiting time and maintains engagement during the audio synthesis phase. The sequential audio playback queue ensured that multi-sentence responses with multiple audio chunks were delivered in correct order without gaps or overlaps.

F. Limitations

Several limitations were identified during evaluation. The Wav2Vec2 SER model is currently stubbed to return neutral for all inputs, pending completion of model caching. Consequently, emotion-conditioned response generation is active at the system prompt level but not driven by real acoustic emotion classification. The LOCAL NIM Audio2Face mode, which would eliminate the cloud gRPC dependency and reduce blendshape latency to approximately one second, is partially implemented but non-functional due to GPU driver conflicts on the Windows host. Detected emotion labels are not yet mapped to facial expression blendshape offsets on the avatar, meaning the avatar currently expresses only speech-driven lip articulation and procedural idle animation without affect-modulated facial expressions such as smiling for happy or frowning for sad. Additionally, conversational memory is bounded to a single hardcoded conversation session, with no user interface for session management.

V. CONCLUSION

This paper has presented EmotionSync, a locally-hosted real-time conversational AI companion integrating speech recognition, emotion classification, retrieval-augmented generation, locally-served large language model inference, neural text-to-speech synthesis, and photorealistic 3D avatar animation within a unified streaming pipeline. By treating the creation of an affective AI companion as a combined multimodal streaming engineering problem rather than an isolated natu-ral language processing task, the proposed system achieves frame-

accurate lip synchronization, emotionally coherent re-sponse generation, and persistent conversational memory on consumer-grade GPU hardware.

The multi-mode Audio2Face client design provides ro-bust degradation across cloud, local NIM, local Omniverse, and energy-based fallback modes, ensuring baseline lip an-imation functionality under all deployment conditions. The performance.now()-anchored blendshape dispatch mechanism eliminates timing drift and delivers sub-frame-accurate lip synchronization without dedicated synchronization hardware. Experimental evaluation confirms that the integrated pipeline operates at latencies suitable for natural conversation, with first audio response delivery within three to four seconds of utterance completion on the target hardware. The system represents a practical demonstration that the building blocks for emotionally intelligent, photorealistic conversational AI are available and integrable today on accessible hardware without proprietary cloud LLM dependencies.

Future work will focus on four primary directions: (1) activating full Wav2Vec2-based SER inference and mapping detected emotions to avatar facial expression blendshape off-sets for affect-driven visual feedback; (2) completing the LO-CAL NIM Audio2Face integration to eliminate cloud gRPC dependency and reduce blendshape latency; (3) implementing conversation summarization to compress older session context and support extended multi-session memory; and (4) extending support to additional languages through multilingual Whisper and compatible neural TTS voices. Longer-term research will explore end-to-end multimodal architectures that jointly model speech prosody, semantic content, and facial animation within a single learned framework.

REFERENCES

1. A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, ”Robust speech recognition via large-scale weak supervision,” in Proc. ICML, 2023.
2. H. Touvron et al., ”LLaMA: Open and efficient foundation language models,” arXiv:2302.13971, Feb. 2023.
3. R. W. Picard, Affective Computing. MIT Press, 1997.
4. H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau, ”Towards em-pathetic open-domain conversation models: A new benchmark and dataset,” in Proc. ACL, 2019.
5. J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, ”Practice and theory of blendshape facial models,” in Eurographics State of the Art Reports, 2014.
6. A. Baevski, H. Zhou, A. Mohamed, and M. Auli, ”wav2vec 2.0: A framework for self-supervised learning of speech representations,” in Proc. NeurIPS, 2020.
7. P. Lewis et al., ”Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Proc. NeurIPS, 2020.
8. N. Reimers and I. Gurevych, ”Sentence-BERT: Sentence embeddings using Siamese BERT-Networks,” in Proc. EMNLP, 2019.
9. Microsoft Corporation, ”Azure Neural Text-to-Speech: en-US-AvaNeural,” Azure Cognitive Services Documentation, 2023.
10. NVIDIA Corporation, ”Audio2Face 3D: Real-time facial animation from audio via NVIDIA ACE,” NVIDIA Developer Documentation, 2023.