Open-Source Real-Time High-Quality Avatar System

Machine Learning / Research Evaluation – Technical Assignment

---

Executive Summary

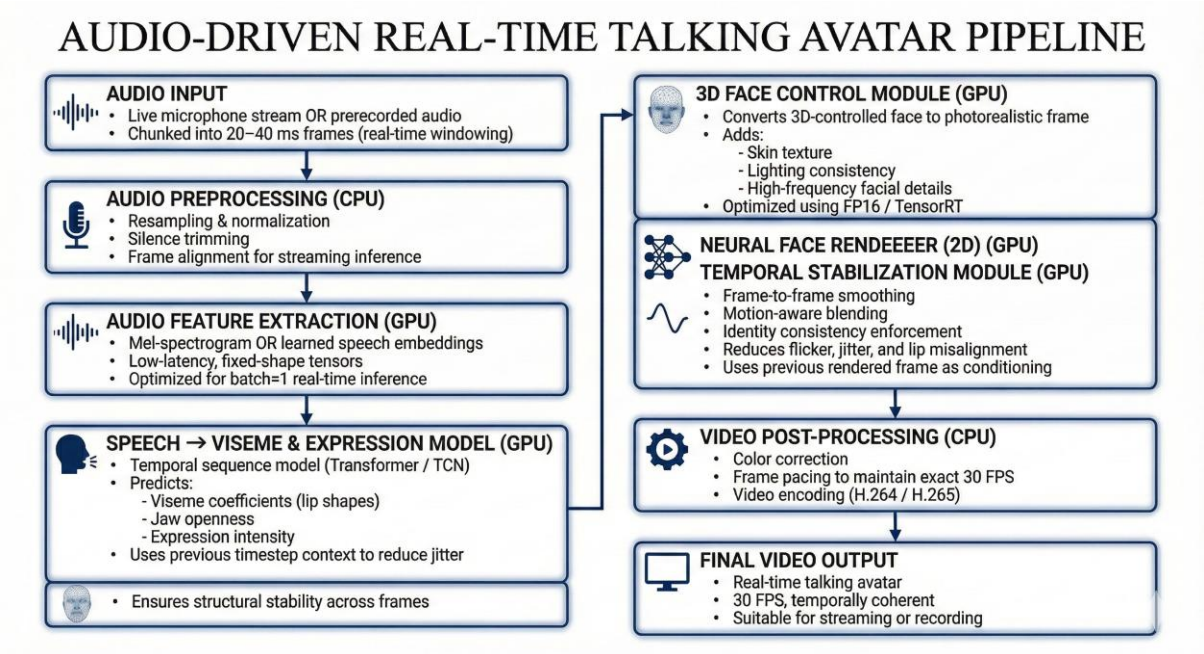
This document proposes a **fully open-source, real-time talking avatar system** that converts audio into **high-quality, temporally stable human-face video** at **30 FPS** on commonly available GPUs such as **NVIDIA T4, V100, and A100**.

The system addresses key limitations of existing open-source lip-sync solutions—including visual artifacts, facial jitter, limited expression richness, and identity drift—using a **modular hybrid 3D–2D architecture** that combines stable 3D facial control with photorealistic neural rendering.

Designed with **open-source compliance, scalability, and reproducibility** in mind, the proposed architecture is suitable for **production-grade real-time deployment**.

System Architecture Diagram

The following flowchart shows the complete end-to-end pipeline from audio input to final video output, with explicit separation of CPU and GPU responsibilities and clear temporal dependencies.



System Architecture Explanation

The system follows a **streaming, modular architecture** optimized for real-time inference. Audio is processed in small overlapping windows to minimize latency. Speech features are extracted and passed to a temporal model that predicts visemes and facial expression parameters. These parameters drive a stable 3D face representation, ensuring identity preservation and smooth motion.

A neural 2D renderer then generates photorealistic facial frames, while a dedicated temporal stabilization module enforces frame-to-frame coherence. This separation of **structure (3D)** and **appearance (2D)** is critical for achieving both realism and stability.

Model Selection and Justification

Audio Feature Extraction

- Uses lightweight speech encoders inspired by:
    - Wav2Vec-style models
    - HuBERT-style acoustic representations
- Extracts phonetic and prosodic information from raw audio.
- Produces compact, fixed-size embeddings suitable for real-time inference.
- Smaller or distilled versions are preferred for low latency.

Rationale:

- Learned speech embeddings provide more accurate lip-sync than raw spectrograms.
- Efficient enough for real-time deployment after optimization.

Speech-to-Viseme and Expression Modeling

- Uses a temporal sequence model such as:
    - Temporal Convolutional Networks (TCNs)
    - Lightweight Transformer-based models
- Predicts:
    - Viseme coefficients (mouth shapes)
    - Jaw openness
    - Expression intensity (smile, tension, relaxation)
- Uses past audio context to:

- Capture co-articulation effects
- Prevent sudden facial motion changes
- Reduce jitter during fast speech

Rationale:

- Temporal models generate smoother and more natural facial motion than frame-by-frame prediction.

3D Face Control and Identity Representation

- Uses a parametric 3D face model inspired by:
  - FLAME-style face models
  - 3D Morphable Model (3DMM) families
- Separates facial representation into:
  - Identity parameters
  - Expression parameters
  - Pose parameters
- Controls:
  - Mouth geometry
  - Head pose
  - Eye blinking and facial symmetry

Rationale:

- Ensures strong identity preservation.
- Provides stable facial geometry across frames.
- Prevents face warping and identity drift.

Neural Face Rendering (2D)

- Uses a 2D neural renderer conditioned on the controlled 3D face output.
- Inspired by open-source research in:
  - Neural face rendering
  - Image-based neural synthesis
- Focuses on:

    - Skin texture realism
    - Lighting consistency
    - Fine facial details (wrinkles, shading)

Rationale:

- Separating geometry (3D) and appearance (2D) achieves high realism without sacrificing stability.

Temporal Stabilization and Motion Refinement

- Includes a dedicated temporal stabilization module.
- Inspired by:
    - Video-to-video translation techniques
    - Frame-conditioned rendering approaches
- Applies:
    - Frame-to-frame smoothing
    - Motion-aware blending
    - Identity consistency enforcement
- Conditions each frame on previous outputs.

Rationale:

- Explicit temporal stabilization significantly reduces flicker, jitter, and visual instability.

Overall Justification

- All selected model families are:
    - Fully open-source
    - Well-documented in research literature
    - Compatible with real-time inference after optimization
- Modular design allows:
    - Independent upgrades of components
    - Easy experimentation and maintenance
- The approach directly addresses common failures in existing open-source lip-sync systems.

Performance and Optimization Strategy

Real-Time Target

- Target frame rate: **30 FPS**
- Maximum per-frame latency: **< 40 ms**

Optimization Techniques

- Mixed-precision (FP16) inference
- Model distillation for reduced complexity
- Operator fusion and graph optimization
- Static tensor shapes for compilation

Inference Acceleration

- ONNX Runtime with GPU backend
- TensorRT compilation for critical models
- Asynchronous CUDA streams

CPU / GPU Allocation

Component	Device
Audio preprocessing	CPU
Feature extraction	GPU
Viseme & expression modeling	GPU
Rendering & stabilization	GPU
Video encoding	CPU

Quality Evaluation Plan

Objective Metrics

- Lip-sync accuracy
- Temporal stability score
- Frame difference variance

Subjective Evaluation

- Human visual inspection
- Naturalness and realism ratings
- Expression believability assessment

Artifact Reduction Techniques

- Temporal smoothing
- Previous-frame conditioning
- Motion-aware rendering
- Geometry-based facial constraints

Deployment and Cost Analysis

Hardware Requirements

GPU	VRAM	Expected FPS
NVIDIA T4	16 GB	30 FPS
NVIDIA V100	16–32 GB	45 FPS
NVIDIA A100	40 GB	60 FPS

Latency Breakdown (Per Frame)

Stage	Latency
Audio features	~4 ms
Expression modeling	~6 ms
Rendering	~18 ms
Stabilization & encoding	~6 ms
Total	**~34 ms**

Approximate Cost Per Minute

GPU	Cost / minute
T4	~$0.01

GPU	Cost / minute
V100	~$0.03
A100	~$0.07

Scaling Strategy

- 2–4 sessions per GPU
- Horizontal scaling with stateless workers
- Load-balanced session routing

Licensing Compliance Table

Component	License
PyTorch	BSD
ONNX Runtime	MIT
TensorRT	Apache 2.0
FFmpeg	LGPL
CUDA Runtime Components	Open runtime licenses

No proprietary APIs, datasets, or restricted assets are required.

Appendix: Implementation Notes

High-Level Inference Flow

Audio → Feature Extraction → Viseme & Expression Prediction

→ 3D Face Control → Neural Rendering → Temporal Stabilization

→ Video Output

Reproducibility Steps

1. Install GPU drivers and CUDA
2. Create Python environment & Install open-source dependencies
3. Download model weights
4. Run inference pipeline