Open-Source Real-Time High-Quality Avatar Generation System

Technical Architecture Proposal

---

1. Executive Summary

This document proposes a fully open-source, production-grade system for real-time generation of high-quality talking-head avatar video from audio input. The system is designed to operate at approximately 30 frames per second on commonly rentable GPUs such as NVIDIA T4, V100, and A100, while prioritizing visual fidelity, temporal stability, and expressive realism.

The proposed architecture addresses common shortcomings observed in existing open-source lip-sync and talking-head systems, including temporal jitter, visual artifacts, identity drift, and limited expression richness. These issues are mitigated through a motion-first design that separates facial motion modeling from image rendering, explicit temporal modeling, and efficient inference-time optimizations.

The system operates using audio-only input at inference time, assuming a pre-initialized avatar identity. Video data is used exclusively during offline training to extract supervision signals and is not required during deployment. All components are designed to comply with permissive open-source licenses that allow commercial use, ensuring reproducibility and practical deployability.

Rather than prescribing a single fixed implementation, this proposal defines a modular, model-agnostic architectural blueprint that balances quality, performance, and open-source compliance, serving as a foundation for scalable real-time avatar systems.

---

2. Problem Definition and Scope

2.1 Problem Definition

The objective is to design an end-to-end system that converts raw audio input into a temporally coherent, visually realistic talking-head avatar video in real time. The system must support continuous audio streams, maintain identity consistency across frames, and generate expressive facial motion that aligns with speech content and prosody.

Existing open-source approaches often rely on frame-independent image generation or direct pixel-space prediction, which leads to unstable motion, flickering artifacts, and degraded quality under real-time constraints. This proposal focuses on addressing these limitations through structured motion representations, temporal modeling, and efficient rendering strategies.

2.2 Assumptions and Scope

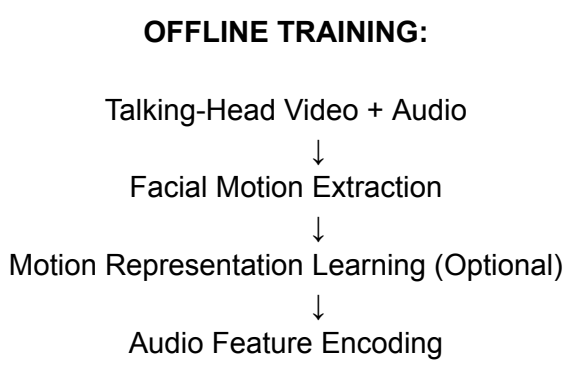
- The system assumes a **pre-initialized avatar identity**, defined via a reference image or identity embedding.
- At inference time, the system operates using **audio-only input**.
- The output is a **head-and-shoulders talking avatar video**; full-body animation is out of scope.
- Background and torso regions are static or minimally animated.
- The system targets **~30 FPS real-time performance** on a single GPU.
- All components must comply with **permissive open-source licenses allowing commercial use**.

---

3. System Architecture Overview

The system is divided into two primary pipelines: an **offline training pipeline** and an **online inference pipeline**. This separation ensures efficient deployment while leveraging video data only where necessary.

3.1 High-Level Architecture



↓
Temporal Audio-to-Motion Model
↓
Motion-Conditioned Renderer Training

ONLINE INFERENCE:

Audio Input
↓
Audio Feature Encoding
↓
Temporal Motion Prediction
↓
Motion Smoothing
↓
Rendering (Identity + Motion)
↓
Video Output (30 FPS)

---

4. Offline Training Pipeline

4.1 Input Data

- Talking-head videos with synchronized audio
- One or multiple speakers
- Video data is used **only for supervision**, not during inference

---

4.2 Facial Motion Extraction (Supervision Creation)

Since facial motion labels are not natively available, a deterministic preprocessing stage is used to extract structured facial motion signals.

Processing Steps

- Face detection and alignment
- Facial landmark extraction
- Head pose estimation
- Eye and mouth state estimation

Per-Frame Motion Representation

```
m_t = {
  jaw_open,
  mouth_width,
  lip_rounding,
  jaw_rotation,
  eye_blink,
  head_pose (pitch, yaw, roll)
}
```

This representation provides a compact, interpretable description of facial motion and is independent of identity.

4.3 Motion Representation Learning (Optional)

To improve robustness and smoothness, an optional motion autoencoder is trained:

```
raw_motion → latent_motion (8–32D)
```

Benefits

- Noise reduction
- Enforced smoothness
- Renderer-friendly representation
- License-clean alternative to parametric face models

4.4 Audio Feature Encoding

Input

- Raw audio waveform

Encoding

- Pretrained self-supervised or signal-based speech encoders (e.g., HuBERT-style or Mel-based)

Output

```
a_t ∈ ℝ^D
```

Pretrained encoders may be frozen or lightly fine-tuned, subject to licensing constraints.

4.5 Temporal Audio-to-Motion Model

This model learns the mapping from audio features to facial motion dynamics.

Input

```
[a_(t-k) … a_t]
```

Target

```
latent_motion(t) or raw_motion(t)
```

Model Options

- Temporal CNN + GRU/LSTM
- Causal Transformer

Training Losses

- Motion regression loss
- Temporal smoothness regularization
- Velocity and acceleration constraints

This stage is responsible for producing temporally coherent, expressive facial motion.

4.6 Renderer Training

The renderer converts identity and motion into photorealistic images.

Inputs

- Static identity representation
- Motion parameters
- Target video frame

Architecture

- Identity encoder (static, frozen during inference)
- Motion-conditioned generator

The renderer may leverage auxiliary signals such as geometric buffers or previous-frame features to enhance temporal stability.

---

5. Online Inference Pipeline

5.1 Identity Initialization

Performed once per avatar.

Input

- Reference image or identity embedding

Output

- Frozen identity vector

5.2 Audio Streaming and Encoding

- Live or recorded audio input
- Audio features extracted in real time

5.3 Temporal Motion Prediction

Input

```
audio_features(t-k … t)
```

Output

```
predicted_motion m^_t
```

The model operates causally to support streaming inference.

5.4 Motion Smoothing

Inference-time smoothing is applied to reduce residual jitter.

Techniques

- Exponential moving average

- Short temporal buffers

5.5 Rendering and Video Assembly

Input

```
(identity, smoothed_motion)
```

Output

- RGB frame at ~30 FPS
- Head-and-shoulders crop
- Static background composition

---

6. Performance and Optimization Strategy

6.1 Real-Time Performance

- Target: ~30 FPS
- Batch size: 1
- Deterministic per-frame latency

6.2 Optimization Techniques

- FP16 inference
- Model pruning and distillation (optional)
- TorchScript / ONNX Runtime / TensorRT

- Pipeline parallelism

6.3 Compute Allocation

- CPU: audio preprocessing, orchestration
- GPU: motion prediction and rendering
- Memory usage optimized for predictable VRAM consumption

---

7. Quality-First Techniques and Evaluation

7.1 Temporal Stability

- Sequence-level training
- Temporal consistency losses
- Motion-space smoothing
- Renderer temporal conditioning

7.2 Expression Richness

- Jaw, lips, eyes, and head pose modeled explicitly
- Prosody-aware motion
- Independent eye blink and head motion generation

7.3 Evaluation Metrics

Objective

- Landmark error
- Lip Motion Distance (LMD)
- Temporal LPIPS

Subjective

- Mean Opinion Score (MOS)
- Side-by-side human evaluation
- Identity preservation assessment

---

8. Deployment and Cost Analysis

8.1 Hardware Requirements

GPU	Estimated VRAM	Streams
T4	8-10 GB	1
V100	12-16 GB	1-2
A100	16+ GB	2-4

8.2 Cost Estimate

- T4: ~$0.35/hour
- Approx. cost: **$0.006–$0.01 per minute per stream**

8.3 Scaling Strategy

- Horizontal scaling
- Session-based GPU allocation
- Optional batching for non-real-time workloads

8.4 Monitoring

- FPS and latency
- Audio-video drift
- Frame variance
- GPU utilization and memory

---

9. Open-Source Compliance and Reproducibility

9.1 Licensing Principles

- Permissive licenses only (MIT, Apache 2.0, BSD)
- No proprietary datasets or assets
- Model-agnostic replacements allowed

9.2 Licensing Table

Component	License Type	Notes
Audio Encoder	Apache/MIT	Commercial-safe
Motion Model	MIT/BSD	Custom trained
Renderer	MIT	Replaceable

Runtime Tools	Apache/BSD	ONNX/TensorRT

9.3 Reproducibility

- Standard GPU environment
- Open-source dependencies
- Modular components
- Clear training vs inference separation

---

10. Optional Appendix

- Prototype experiments
- Pseudo-commands
- Implementation notes
- Future improvements

Final Note

This proposal defines a production-ready architectural blueprint for real-time avatar generation that balances quality, performance, and open-source compliance. The modular design enables iterative improvement while remaining deployable under real-world constraints.