StreamAvatar: Real-Time-Capable Audio-Driven Portrait Animation via DyStream Optimization and Blockwise Motion Distillation

PanChangxun GuoZihan LiSi

ABSTRACT

We propose StreamAvatar, a real-time-capable audio-driven portrait animation system built on top of DyStream, a pretrained autoregressive flow-matching talking-head model. The project targets a practical course-scale question: can a strong offline-quality talking-head teacher be optimized and distilled into a lower-latency streaming system without training a full portrait renderer from scratch? Our plan is to keep DyStream's pretrained audio encoder, motion prior, and renderer as the teacher path, optimize its inference runtime, and train a blockwise student that predicts short future motion-latent blocks from audio, rolling history, and reference anchors. The expected outcome is a browser-facing prototype that can animate a reference portrait from audio, together with quantitative analysis of speed, lip synchronization, motion strength, and anchor conditioning. We will evaluate with SyncNet/LSE metrics, landmark-based mouth statistics, latent motion losses, qualitative videos, and measured real-time factor.

1 Introduction

Audio-driven talking portrait generation synthesizes a video of a reference person speaking according to an input audio stream. A useful system must preserve identity, synchronize lips to speech, avoid temporal collapse, and run fast enough for interactive use. Recent systems show that high-quality talking-head animation is possible, but many approaches remain expensive because they rely on sequential autoregressive generation, flow matching, diffusion-like sampling, or full-frame rendering at inference time.

StreamAvatar focuses on the deployable middle ground. Instead of training a new video generator from scratch, we use DyStream [1] as a pretrained teacher. DyStream already decomposes the problem into audio encoding, autoregressive motion generation, flow-matching motion sampling, and portrait rendering. This decomposition makes it attractive for real-time research: appearance rendering and identity preservation can remain frozen, while the project concentrates on runtime optimization and audio-to-motion distillation.

The core hypothesis is that a blockwise student can approximate the expensive teacher motion rollout with far fewer sequential operations. If the student predicts short future motion blocks directly from audio, motion history, and a reference anchor, it can reduce motion-generation cost while keeping the pretrained renderer unchanged. The project therefore studies not only whether the system can run in real time, but also what quality is lost when replacing autoregressive flow-matching rollout with direct blockwise prediction.

2 Project Goals

The main goal is to build and evaluate a real-time-capable StreamAvatar prototype. Given a reference portrait and an audio clip or microphone stream, the system should output a talking portrait video with stable identity and acceptable lip synchronization.

Preprint. Under review.

The planned contributions are:

- Deploy the DyStream teacher locally and profile its inference bottlenecks.
- Optimize the teacher runtime with model warm-up, cached reference preprocessing, chunked audio handling, batched rendering, and stage-level timing.
- Build a browser-facing prototype with upload or microphone input, backend inference, frame queuing, and continuous playback.
- Train a blockwise student model that predicts future motion-latent blocks conditioned on audio, rolling motion history, and reference anchors.
- Compare real cached anchors, synthetic noise anchors, and mixed-anchor training as conditioning strategies.
- Evaluate speed-quality trade-offs using synchronization, landmark, latent, and qualitative metrics.

We do not aim to train a high-resolution portrait renderer or outperform the teacher in visual fidelity. The intended contribution is a practical optimization and distillation study for streaming audio-driven portrait animation.

3 Method

3.1 Teacher System

DyStream [1] provides the teacher. Its frozen components include a wav2vec-style audio frontend [2], an autoregressive GPT-like motion generator, a flow-matching motion sampler [3], and a pretrained renderer. For an audio sequence and reference image, the teacher generates a sequence of motion latents and renders them into video frames.

The teacher path serves two purposes. First, it is the quality baseline for generated videos. Second, it produces pseudo-target motion trajectories for student distillation. Keeping the teacher frozen reduces training cost and makes the project feasible with limited data and time.

3.2 Runtime Optimization and Streaming Prototype

The first engineering stage is to make DyStream usable as a local application. We will move model loading out of the hot path, preprocess the reference image once, cache reusable tensors, batch rendering when possible, and expose per-stage timing. For streaming-style playback, audio is processed in chunks, predicted or rendered frames are placed into a queue, and the frontend consumes frames at the target frame rate.

The prototype is expected to be real-time-capable rather than a fully optimized video-call system. Live latency depends on audio chunk size, model execution, rendering, muxing or frame transport, and playback buffering. We will therefore report both offline throughput and streaming behavior.

3.3 Blockwise Motion Student

The student is designed to replace the expensive teacher motion rollout. For each block, it receives audio features, a rolling motion history, a reference anchor, noisy future-token input, and a timestep. It then predicts a clean future motion block:

$$\hat{\mathbf{M}}_{t:t+B-1} = f_\theta(\mathbf{A}_{t:t+B-1}, \mathbf{M}_{t-H:t-1}, \mathbf{r}, \mathbf{Z}_{t:t+B-1}, \tau), \tag{1}$$

where B is the block size, H is the history length, $\mathbf{A}$ denotes audio features, $\mathbf{r}$ is the reference anchor, $\mathbf{Z}$ denotes noisy future tokens, and τ is the sampled timestep. During rollout, the predicted block is appended to the history before predicting the next block.

The retained flow-matching-style interface is used as a denoising input, not as a full ODE solver. The student performs one forward pass per block and directly predicts clean motion. This should provide a clear speed advantage over sequential teacher rollout.

3.4 Training Losses

The student is trained with supervised motion losses against teacher-generated or cached real motion targets:

$$\mathcal{L}_{\text{motion}} = \|\hat{\mathbf{M}} - \mathbf{M}^{\star}\|_2^2. \tag{2}$$

We will also include velocity, acceleration, and boundary losses:

$$\mathcal{L} = \mathcal{L}_{\text{motion}} + \lambda_v \mathcal{L}_{\text{velocity}} + \lambda_a \mathcal{L}_{\text{acceleration}} + \lambda_b \mathcal{L}_{\text{boundary}}. \tag{3}$$

Velocity and acceleration terms discourage slow-motion collapse and jitter. The boundary term encourages adjacent predicted blocks to connect smoothly.

3.5 Anchor Conditioning

The reference anchor is a 512-dimensional latent extracted from a reference image or cached training clip. We will compare three regimes:

- **Real anchors**: cached anchors extracted from LRS3 motion latents.
- **Noise anchors**: synthetic Gaussian or sphere samples.
- **Mixed anchors**: a mixture of real and synthetic anchors.

This study tests whether synthetic anchors are useful as data augmentation or whether they fail to match the real anchor manifold. Distribution plots in raw dimensions and PCA space will be used to interpret the result.

4 Experiments

4.1 Data

We will use LRS3 [4] video/audio clips. Training examples are short segments with synchronized audio and motion latents. Two target sources are planned: teacher-generated motion trajectories and cached real motion latents extracted from videos. A held-out validation split will be used to monitor overfitting and compare anchor strategies.

4.2 Evaluation Metrics

The evaluation will include:

- SyncNet LSE-D and LSE-C for audio-video synchronization [5, 6].
- Landmark-based mouth opening correlation and mouth velocity.
- Latent motion MSE/L1 between student and teacher or cached-real targets.
- Velocity ratio between student and teacher motion.
- Runtime, real-time factor, and student-vs-teacher motion rollout speedup.
- Qualitative side-by-side videos comparing teacher, real-anchor student, and mixed-anchor student.

4.3 Ablations

The planned ablations are:

- Teacher path versus student path.
- Real-anchor student versus noise-anchor or mixed-anchor student.
- Blockwise rollout with and without auxiliary velocity, acceleration, and boundary losses.
- Different block sizes and history lengths, subject to training stability.

5 Expected Outcomes

We expect the optimized teacher path to approach the real-time boundary on a modern GPU after warm-up and batching, while the student should provide a larger speedup specifically for motion rollout. We also expect the teacher to remain the best visual-quality reference. The key research outcome is the measured trade-off: how much synchronization, mouth-motion strength, and identity consistency are retained when replacing AR+FM teacher rollout with blockwise student prediction.

For anchor conditioning, the working hypothesis is that real cached anchors will be the safest default because they lie on the training distribution. Synthetic anchors may regularize identity or motion strength, but naive high-dimensional noise may not faithfully cover the true anchor manifold. PCA or covariance-aware sampling is a likely future improvement if simple noise anchors are insufficient.

6 Timeline

Stage	Milestone
1	Deploy DyStream teacher, verify checkpoints, and generate baseline videos.
2	Add runtime profiling, cached preprocessing, batched rendering, and local app controls.
3	Build LRS3 cache for audio and motion latents.
4	Train and debug blockwise student overfit runs, then scale to cached data.
5	Run anchor-conditioning experiments and generate qualitative comparisons.
6	Evaluate SyncNet, landmark, latent, and speed metrics; write final report.

7 Conclusion

StreamAvatar proposes a practical route toward real-time audio-driven portrait animation: keep the strong pretrained DyStream renderer and teacher, optimize the runtime, and distill the expensive motion rollout into a blockwise student. The project is intentionally scoped around measurable engineering and modeling questions rather than full video-generator training. Its success will be judged by a working prototype, clear speed-quality measurements, and an analysis of when student distillation and anchor conditioning help or hurt.

References

[1] Ruobing Li et al. Dystream: Streaming dyadic talking heads generation via flow matching-based autoregressive model. *arXiv preprint*, 2025.

[2] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, 2020.

[3] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Proceedings of the International Conference on Learning Representations*, 2023.

[4] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. Lrs3-ted: A large-scale dataset for visual speech recognition. In *arXiv preprint arXiv:1809.00496*, 2018.

[5] Joon Son Chung and Andrew Zisserman. Out of time: Automated lip sync in the wild. In *Proceedings of the Asian Conference on Computer Vision Workshops*, 2016.

[6] K. R. Prajwal, Rudrabha Mukhopadhyay, Vinay P. Namboodiri, and C. V. Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the ACM International Conference on Multimedia*, 2020.