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

Pan Changxun
Institute for Interdisciplinary Information Sciences
Tsinghua University
pancx24@mails.tsinghua.edu.cn

Guo Zihan
Institute for Interdisciplinary Information Sciences
Tsinghua University
guo-zh24@mails.tsinghua.edu.cn

Li Si
Institute for Interdisciplinary Information Sciences
Tsinghua University
s-li24@mails.tsinghua.edu.cn

Project page: https://cxp-2024.github.io/StreamAvatar/

ABSTRACT

StreamAvatar is a real-time-capable audio-driven portrait animation system built by adapting and accelerating DyStream, a pretrained autoregressive flow-matching talking-head model. This report describes the resulting system. We deploy the DyStream teacher, optimize its inference path, provide an offline Gradio demo and project page, and train AROD, an Autoregressive One-step Denoising student that predicts short future motion blocks conditioned on audio, history, noisy future tokens, and reference anchors. We further study cross-attention and one-step denoising student variants, real-anchor versus synthetic-anchor conditioning, and supervised training from cached LRS3 motion latents. The optimized teacher path can approach the real-time boundary on a B20Z/B200-class GPU after warm-up, while AROD provides roughly $8\times$ faster motion inference in a fresh 60-second verification run and has reached about $10\times$ in the same verification setup. Our experiments show that real cached anchors give the most stable objective behavior, while mixed noise anchors sometimes improve visible motion strength and long-horizon identity consistency but do not reliably improve SyncNet metrics. The qualitative Person2 comparison is rendered as a 60-second three-way video and remains visually identity-consistent across the full long-horizon sequence. The final outcome is a StreamAvatar offline demo, a public project page, and a streaming-capable blockwise inference design, with clear empirical lessons about distilling autoregressive flow-matching portrait animation models.

1 Introduction

Audio-driven talking portrait generation synthesizes a video of a reference person or avatar speaking according to an input audio stream. It is useful for virtual meetings, low-bandwidth telepresence,

Preprint. Under review.

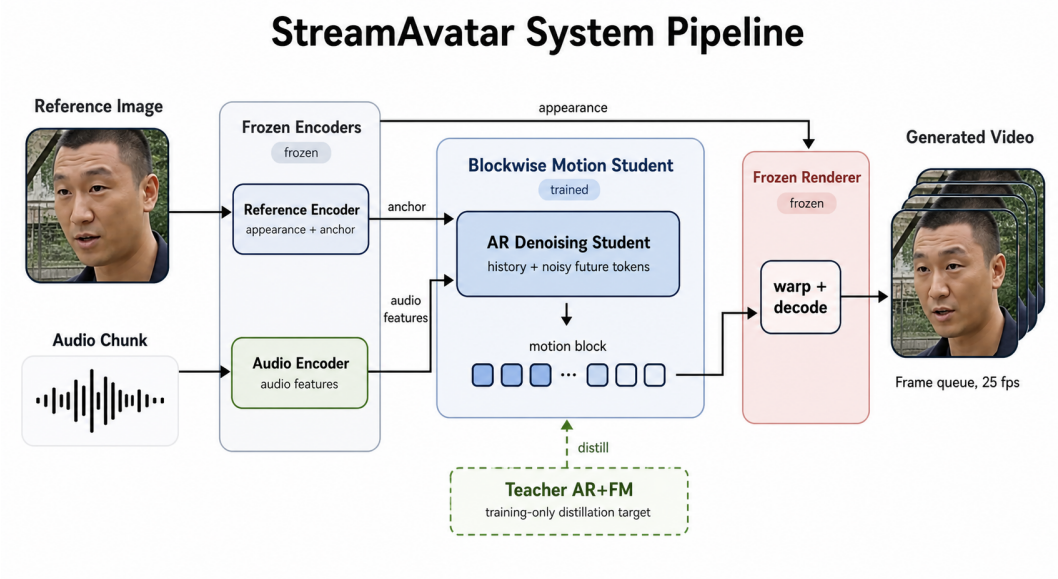


Figure 1: Final StreamAvatar system. The original DyStream teacher is preserved as the quality reference. Our student path bypasses the expensive AR+FM teacher rollout by predicting short motion blocks directly from audio, history, and anchors.

privacy-preserving representation, and avatar-based communication. A practical system must satisfy three constraints at the same time: the lips should match the speech, the identity should remain stable over long clips, and inference should be fast enough for interactive use.

StreamAvatar is designed around a practical observation: a strong pretrained talking-head model already contains most of the visual prior needed for high-quality portrait animation. DyStream [1] provides the key components: a wav2vec-style audio front-end [2], an autoregressive GPT-like motion generator, a flow-matching motion sampler [3], and a pretrained renderer. Instead of training a full portrait generator from scratch, we treat DyStream as a teacher and study how far it can be optimized, streamed, and distilled into a cheaper audio-to-motion student.

This report focuses on the implemented system. Our contribution is an engineering and modeling study of real-time-capable audio-driven portrait animation:

- We deploy DyStream locally and analyze its inference bottlenecks.
- We build an optimized offline inference demo and analyze the requirements for extending the blockwise path to streaming playback.
- We train blockwise student models that predict future motion blocks directly from audio, history motion, and reference anchors.
- We compare real cached anchors, synthetic anchors, and mixed-anchor training, including PCA/t-SNE distribution analysis.
- We evaluate with SyncNet/LSE metrics [4, 5], landmark-based mouth motion statistics, loss curves, qualitative videos, and speed measurements.

The main result is pragmatic. We do not claim to outperform the teacher in visual quality or to deliver a fully live video-call system. Instead, we show that the teacher path can be made practical as an offline demo and that a compact blockwise student can substantially reduce motion-generation cost, at the price of some lip-sync and motion fidelity.

2 System Overview

DyStream is naturally modular. The model does not generate pixels directly from audio. Instead, audio is encoded into temporal features, an autoregressive motion model predicts a latent motion trajectory, flow matching refines or samples the motion latent, and a pretrained renderer turns the motion into video frames. This is important for real-time work because the expensive pixel-level identity and appearance prior can remain frozen.

In our optimized runtime, the teacher path is used in two ways. First, it is the baseline used for qualitative comparison. Second, it provides pseudo-targets for distillation: for a fixed audio chunk and reference anchor, the teacher produces a motion trajectory, and the student learns to produce a similar trajectory with fewer sequential operations.

3 Method

3.1 Optimized DyStream Inference

We first made the DyStream inference path usable as a local application. The important optimizations were not all model-level. Model loading is moved out of the hot path, the reference image is preprocessed once, rendering is batched, and the web app reports stage-level timing. The best practical setting used a small motion chunk size and batched rendering. In our profiling, the original teacher path remains close to the real-time boundary rather than being an order of magnitude faster: a 60-second baseline run was dominated by motion generation and rendering, while warmed-up short-clip tests were around real-time. The large speedup reported later therefore refers to the distilled student motion rollout, not to the full DyStream teacher pipeline.

The current application interface is an offline Gradio demo: a user provides a reference image and an audio file, the backend predicts the full motion sequence, and the renderer writes the output video. A live version would require an additional frame queue, microphone noise handling, browser-side scheduling, and rolling backend state. We treat these as streaming-system requirements rather than claims about the current delivered app.

3.2 Blockwise Student Distillation

The teacher uses an autoregressive structure. A GPT-like model conditions on previous motion and audio, then the flow-matching module generates motion latents. This is high quality but sequential. We call our final student **AROD**, short for *Autoregressive One-step Denoising*. AROD should be understood as a blockwise autoregressive denoising model, not as the teacher's two-stage AR+FM pipeline. For each block, it receives rolling motion history, audio features, a reference anchor, noisy future tokens, and a timestep, then directly 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 $B = 2$ in our conservative final student, H is the history length, $\mathbf{A}$ denotes audio features, $\mathbf{r}$ is the reference anchor, $\mathbf{Z}$ denotes noisy future-token input, and τ is the sampled timestep. During rollout, the predicted block is appended to history and used by the next block.

We explored several student architectures. The most useful version is AROD's blockwise cross-attention denoising student. History tokens and noisy future tokens are processed in one Transformer sequence. Audio and anchor features are injected through cross-attention. The retained "FM-like" part is only the noise/timestep interface: during training we create a noisy version of the target future block, and during inference we feed Gaussian future noise at the first scheduler timestep. The student does not run an ODE solver and does not predict a velocity field; it directly predicts clean motion in one forward pass per block.

3.3 Training Objectives

The student is trained by rollout. For a training clip, we run the student block by block. Each predicted block is appended to the history for the following block. Gradients are accumulated through the rollout inside one forward/backward step, with optional detachment between blocks for stability.

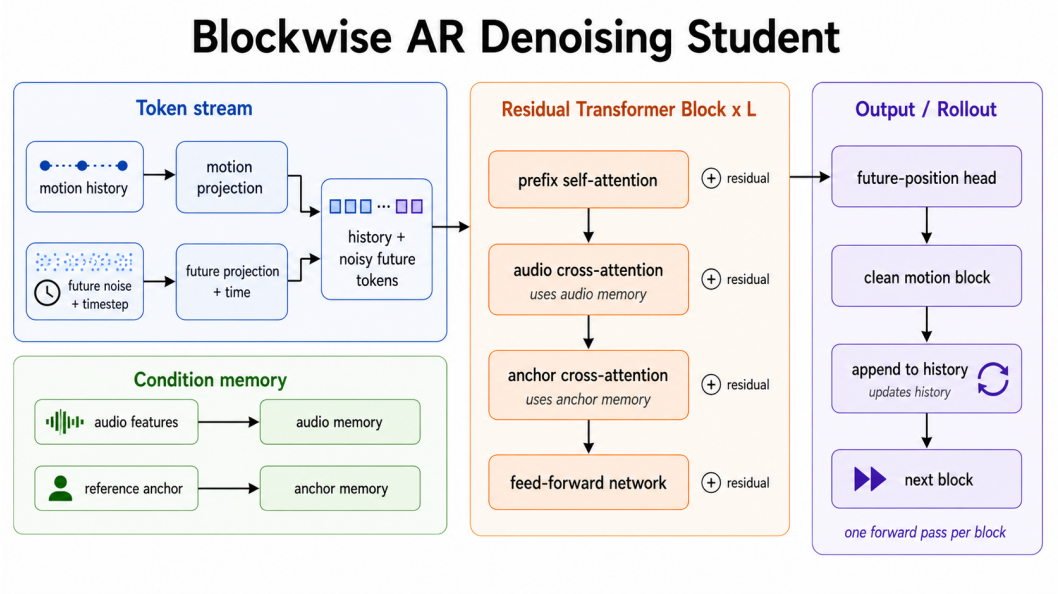


Figure 2: Blockwise AR denoising student. Noisy future tokens are part of the Transformer input, not a separate ODE sampler. The model performs one forward pass per block, outputs the clean future motion block, appends it to history, and then predicts the next block.

The main loss compares the predicted motion block against either teacher motion or cached real motion:

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

We additionally use velocity, acceleration, and boundary losses:

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

The velocity and acceleration losses discourage slow-motion collapse and unnatural jitter. The boundary loss makes adjacent predicted blocks connect smoothly.

3.4 Anchor Conditioning

The reference anchor is a 512-dimensional motion/identity-style latent extracted from the reference image or cached training video. We tested three anchor regimes:

- **Real anchor**: use cached real anchors from LRS3 motion latents.
- **Noise anchor**: use synthetic Gaussian or sphere noise anchors.
- **Mixed anchor**: use real anchors for half the batch and synthetic anchors for the other half.

The mixed-sphere variant samples

$$\mathbf{r}_{\text{noise}} = \bar{\mathbf{r}}_{\text{real}} + R \cdot \frac{\epsilon}{\|\epsilon\|_2}, \quad \epsilon \sim \mathcal{N}(0, I), \tag{4}$$

with $R = 6$. This was intended to cover the real-anchor distribution. However, later PCA analysis showed that high-dimensional sphere sampling does not expand along the true principal directions of real anchors. It is therefore better interpreted as a regularizer than as faithful distribution coverage.

4 Experiments

4.1 Data and Setup

We use LRS3 [6] video/audio clips. Two kinds of targets are used. Teacher-distillation targets are generated online by the DyStream teacher. Cached-real targets are generated by preprocessing LRS3

Model	LSE-D ↓	LSE-C ↑	Mouth corr. ↑	Mouth vel.	Motion MSE ↓	Vel. ratio
Teacher	**7.679**	**4.987**	**0.305**	**0.0346**	–	–
Real-anchor student	8.853	**4.054**	0.225	0.0276	**0.0257**	0.678
Noise+real student	**8.721**	4.036	**0.238**	**0.0305**	0.0277	**0.774**

Table 1: Person1 60-second comparison. Teacher is best overall. Mixed anchors increase visible motion strength and improve velocity ratio, while real anchors better match teacher latents.

Model	LSE-D ↓	LSE-C ↑	Mouth corr. ↑	Mouth vel.	Motion MSE ↓	Vel. ratio	Speedup
Teacher	**7.080**	**6.235**	0.051	0.0326	–	–	–
Real-anchor student	**8.309**	**4.027**	0.0328	0.0336	**0.0305**	0.760	**8.47**×
Noise+real student	8.429	3.280	**0.0789**	**0.0359**	0.0384	**0.832**	7.71×

Table 2: Person2 60-second comparison. Real anchors are better on SyncNet and latent matching; mixed anchors produce stronger mouth motion. The corresponding three-way rendered video remains visually identity-consistent over the full long-horizon clip.

videos into audio waveforms and 512-dimensional motion-latent sequences. For the largest student runs, we used a 60k pretrain-cache manifest with a held-out validation set. Most clips are sampled as 3-second training segments to keep training efficient and compatible with blockwise rollout.

The renderer and teacher components remain frozen. The student is trained with batch size 64 in the main cache/anchor experiments. Validation is run periodically and logged to JSONL. All reported qualitative comparisons use the same audio/reference inputs for teacher and student.

4.2 Metrics

We report the following metrics:

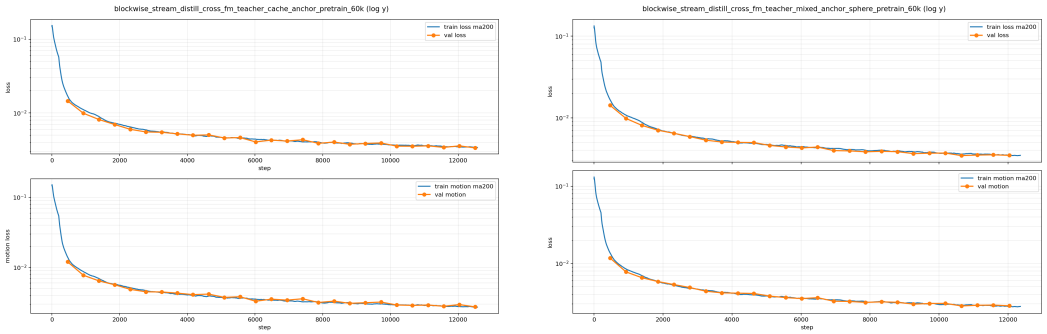
- **LSE-D**: SyncNet audio-video distance; lower is better.
- **LSE-C**: SyncNet confidence; higher is better.
- **Mouth-audio correlation**: correlation between landmark mouth opening and audio energy.
- **Mouth velocity**: average landmark mouth-opening velocity, used as a rough motion-strength indicator.
- **Motion MSE/L1**: latent distance between student and teacher motion.
- **Velocity ratio**: student/teacher motion velocity magnitude. Values closer to one indicate less under-motion.
- **Speedup**: measured student motion rollout speed relative to the teacher rollout in verification.

4.3 Main Quantitative Results

The teacher remains the quality reference. Between students, the result is mixed. Real-anchor training gives more conservative and stable teacher-latent imitation. Noise+real mixed anchors can improve visible mouth motion strength and velocity magnitude, but the SyncNet improvement is not consistent across reference images. Qualitatively, the 60-second Person2 three-way comparison keeps the same facial identity and appearance traits across the long clip, which is important for practical avatar use: the student does not collapse into a different identity after extended autoregressive rollout.

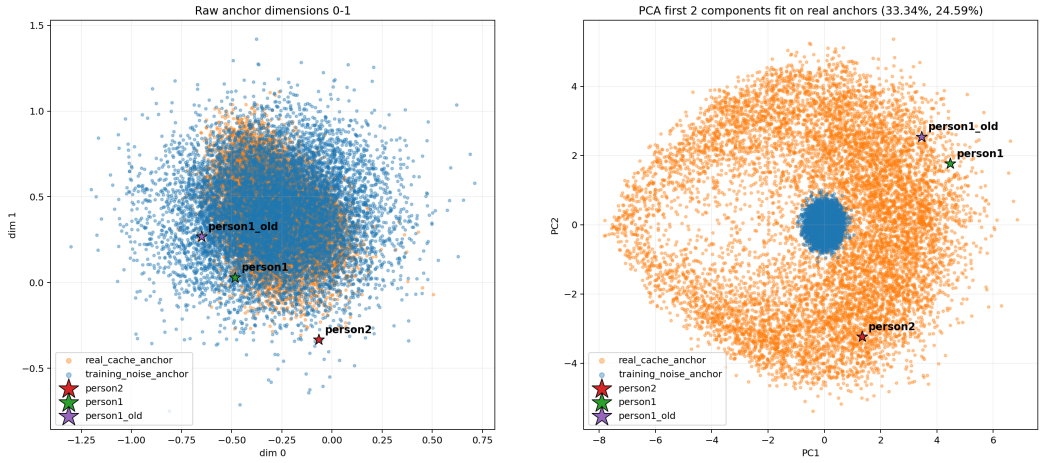
4.4 Training Curves and Anchor Analysis

The anchor study produced a nuanced result. In the original coordinate dimensions, synthetic sphere anchors can appear to cover the target references, so they are a reasonable form of anchor-space data



Real-anchor student Mixed-anchor student

Figure 3: Training curves. Both variants converge smoothly. Real-anchor training is the safer default, while mixed anchors act as regularization rather than a consistently better objective.



Raw anchor dimensions 0–1 PCA fit on real anchors

Figure 4: Two views of anchor augmentation. In the raw coordinate slice, the synthetic sphere anchors broadly cover the selected reference anchors. In the real-anchor PCA view, however, the same samples concentrate near the center of the dominant real-anchor directions. The sphere therefore acts as a useful condition-space augmentation, but it is not a faithful replacement for the real anchor manifold.

augmentation. However, when PCA is fit on the real-anchor distribution, the synthetic sphere anchors are concentrated around the center of the first principal components. In 512 dimensions, large-radius random directions do not necessarily align with the low-dimensional real motion/identity manifold. This is why mixed noise anchors should be reported as an ablation and regularizer, not as the final default training strategy.

4.5 Speed and Streaming Behavior

The optimized teacher pipeline is close to the real-time boundary on a B20Z/B200-class GPU, especially after warm-up and with batched rendering, but it should not be interpreted as an order-of-magnitude faster-than-real-time system. The larger acceleration comes from AROD, which replaces the expensive teacher motion rollout with direct blockwise one-step denoising. In a fresh verification run with ‘verify_blockwise_distill.py’ on a 60.03-second ‘person1 + test_audio_60s.wav’ case, the teacher motion rollout took 29.38 seconds while AROD took 3.75 seconds, giving a 7.83× motion-generation speedup. A separate motion-only benchmark on the same case measured 26.01 seconds for the teacher and 3.25 seconds for AROD, or 8.00×. Earlier verification of the same configuration recorded 10.24×, so we describe the result as approximately 8× in the latest fresh run and capable of reaching about 10× under the same setup, rather than a fixed deterministic 10× for every run.

Figure 5: Qualitative comparison frame from the 60-second Person2 three-way video: teacher, real-anchor student, and noise+real-anchor student. In this example, the mixed-anchor student produces a more conservative head pose than the teacher and preserves facial details such as forehead wrinkles more clearly, although this advantage is not consistent across all metrics. The full rendered sequence is used to check long-horizon identity consistency, and a 3-second preview is shown on the project page.

A real streaming demo also needs warm-up, rolling audio context, rolling motion history, frame buffering, stable playback, and a low-latency browser transport. Our current implementation should therefore be read as an offline Gradio demo plus a streaming-capable model design: the AROD student predicts short future blocks autoregressively, so it can be extended to rolling audio windows without changing the core motion model.

The largest remaining gap is interaction latency. Even when throughput is near real time, live streaming latency is bounded by audio chunk duration, backend processing time, render time, and playback buffering. Our current system is best described as *real-time-capable* or *streaming-capable*, not a fully optimized live streaming or video-call application.

5 Discussion

What worked. The strongest practical success is the decomposition. Keeping the DyStream renderer and teacher frozen makes the task tractable. Optimizing the runtime gives a fast demo. Distilling motion blocks gives a clear speed-quality trade-off. The blockwise denoising student also avoids the worst failure of plain regression: long-horizon identity collapse and severe slow-motion behavior.

What did not fully work. Pure cached-real supervised training from LRS3 motion latents can learn stable motion but may miss behaviors such as natural blinking. This suggests that teacher-generated motion contains useful priors beyond per-frame encoded ground truth. Synthetic anchor training also did not reliably improve objective metrics. It can regularize identity, but simple high-dimensional Gaussian or sphere noise does not match the real-anchor manifold.

Why not simply use one-step teacher FM? The original teacher can run with one denoising step, but the AR GPT component remains a bottleneck. Our student attacks that bottleneck by replacing sequential AR+FM rollout with direct blockwise motion prediction. This is why the student can be faster even when the teacher FM itself is already cheap.

Current best recommendation. For the final demo and report, the real-anchor student should be treated as the primary student model. The mixed-anchor model should be included as an ablation because it reveals a useful phenomenon: anchor perturbation can affect long-term identity consistency and mouth-motion strength, but naive noise does not consistently improve lip-sync metrics.

6 Limitations and Future Work

The current evaluation uses a small number of reference images and audio files. A more reliable conclusion needs a multi-speaker benchmark with averaged SyncNet, landmark, identity, and temporal metrics. The current web app is an offline generation demo. A true live streaming version would need lower-latency audio transport, rolling server-side scheduling, stronger front-end frame scheduling, and better handling of browser microphone noise.

Model-wise, the next step is to replace naive synthetic anchors with distribution-aware anchors. A promising direction is PCA/covariance sampling:

$$\mathbf{r}_{\text{syn}} = \bar{\mathbf{r}} + U_k \Sigma_k^{1/2} \epsilon, \tag{5}$$

where U_k and Σ_k come from real cached anchors. This would sample along the real anchor manifold instead of random 512D directions. Another direction is hybrid teacher/cache training: teacher targets preserve natural motion priors such as blinking, while cached-real targets improve grounding to actual LRS3 data.

7 Team Contribution

Pan Changxun led the core AROD student design, training/inference integration, speed verification, and final system assembly. Guo Zihan led data preprocessing, cached motion-latent preparation, asset organization, and reproducibility checks. Li Si studied related adaptation directions including Chinese fine-tuning, helped compare alternative modeling choices, and contributed to evaluation, report writing, and demo presentation. All members jointly discussed the method, reviewed qualitative results, and polished the final report, website, and release materials.

8 Conclusion

StreamAvatar demonstrates a DyStream-based optimization and distillation path for real-time-capable audio-driven portrait animation. We successfully deployed and accelerated the teacher, built an offline AROD demo, trained blockwise motion students, and analyzed anchor conditioning. The results show that high-quality pretrained talking-head models can be pushed toward streaming-capable use without training a full renderer, but also that distilling autoregressive flow-matching behavior is not a trivial regression problem. Real anchors are currently the safest training condition; mixed noise anchors are useful as an ablation and regularizer but not a clear improvement. The project therefore provides both a working offline demo path and a set of lessons for future real-time avatar research.

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] 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.

[5] 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.

[6] 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.