Early Speech-Like Audio Emergence in a Small OLMo-Hybrid Speech Codec Language Model

A 21.9M-Parameter Pilot on LJ Speech

Micheal Ohagwu
obioh@icloud.com

March 2026

Abstract

We study whether a small OLMo-Hybrid-style recurrent-attention language model can learn enough speech codec structure to generate clearly speech-like audio, without text conditioning, on a clean single-speaker corpus. The model operates on 8-codebook EnCodec 24 kHz tokens extracted from LJ Speech and uses an 8-layer decoder with a 3:1 recurrent-to-attention schedule, yielding 6 Gated DeltaNet blocks and 2 attention blocks for a total of 21.9M parameters. On an NVIDIA A100-SXM4-80GB, the best locally preserved checkpoint reached an EMA validation loss of 4.2847 and perplexity 72.58 at step 1800, before pod failure interrupted the run near the step-2000 boundary. Direct listening to fixed-prompt local samples from steps 1200, 1400, 1600, and 1800 revealed a clear speaking timbre and speech-like local structure, with failures dominated by babble and weak articulation rather than static, drone artifacts, or codec collapse. The stable run did not use the intended fused Flash Linear Attention Gated DeltaNet kernel because of upstream Triton kernel failures, so the reported systems performance should be interpreted as conservative relative to the architecture's intended fast path. We present this as a pilot study rather than a benchmark claim: the main result is that an OLMo-Hybrid / Gated-DeltaNet-style backbone appears to be a viable speech codec language model at small scale.

1 Introduction

Recent open work has renewed interest in hybrid backbones that mix attention with modern recurrent or state-space layers, rather than treating transformers and recurrence as mutually exclusive design choices [6, 8]. In parallel, work on neural audio codecs and token language models has shown that discrete audio tokens are a practical interface for speech and audio generation [1, 3, 7, 9]. A natural next question is whether these ideas combine well: can a small hybrid recurrent-attention backbone model speech codec tokens effectively enough to generate clearly speech-like samples?

This report documents a deliberately narrow pilot designed to answer that question. We did not attempt to build a full text-to-speech system, a streaming agent, or a production voice model. Instead, we focused on a threshold question:

> *Can a compact OLMo-Hybrid-style decoder trained unconditionally on speech codec tokens cross the boundary from static or degenerate decoder output into clearly speech-like audio?*

That threshold matters because it separates architecture and pipeline viability from final product quality. If the model cannot produce anything beyond hiss, drone artifacts, or obvious tokenization failure, then text conditioning, speaker conditioning, and larger-scale training are premature.

If it can produce speech-like babble at small scale, then the architecture is at least plausible for more serious follow-up work.

Our main result is modest but meaningful: on LJ Speech [4], a 21.9M-parameter OLMo-Hybrid-style speech codec language model trained on EnCodec 24 kHz tokens [3] produced clearly speech-like audio by step 1800 on an A100, despite (i) the absence of text conditioning, (ii) the absence of the intended fused recurrent FLA kernel, and (iii) termination well before the planned 12k-step budget. To our knowledge, public speech token language modeling work has focused primarily on transformer-heavy or pure state-space backbones [1, 5, 7, 9]; we are not aware of a prior public pilot centered on an OLMo-Hybrid / Gated-DeltaNet-style backbone for unconditional speech codec language modeling.

We therefore position this document as a technical report rather than a benchmark paper. Its contributions are:

1. a concrete OLMo-Hybrid-style adaptation for speech codec language modeling, described at the level of block schedule, head geometry, token interface, and loss;
2. a small but clean A100 pilot showing monotonic validation improvement through step 1800, reaching EMA validation loss 4.2847 and perplexity 72.58;
3. qualitative evidence that the model learned speech-like local structure well before full convergence; and
4. an honest systems account of what did and did not work, including the stable no-FLA A100 path and the operational consequences of pod volatility.

Representative audio samples and the preserved evaluation history are released alongside this report on the accompanying project page.[1]

2 Related Work

Hybrid recurrent-attention language models. OLMo Hybrid argues that models mixing attention and linear RNN layers are not merely inference-efficient alternatives to transformers, but a broader family that can remain expressive while scaling effectively in pretraining [6]. Their released 7B model uses a 3:1 schedule of Gated DeltaNet and attention layers. The underlying recurrent operator, Gated DeltaNet, extends DeltaNet-style recurrence with gated parameterization, short causal convolutions, and a training-oriented formulation intended to pair with fused Flash Linear Attention kernels [8].

Speech token language modeling. AudioLM established that language models over discrete audio tokens can generate coherent speech and other audio without relying on text supervision [1]. VALL-E reframed zero-shot TTS as neural codec language modeling and showed that codec-token autoregression can support high-quality conditional speech synthesis at scale [7]. SpeechTokenizer argued that tokenizers designed specifically for speech language modeling can outperform more generic alternatives and paired them with a unified speech language model [9]. On the state-space side, DuplexMamba explored Mamba-based speech interaction, but in a duplex streaming speech-to-text conversational setting rather than an unconditional speech codec language model [5].

[1] https://research.nani-inc.com/2026/03/09/olmo-hybrid-ljspeech-a100-pilot.html

Table 1: Architecture and training configuration for the stable A100 run.

Item	Value
Total parameters	21,904,648
Embedding parameters	3,154,944
Backbone parameters	15,594,376
Output-head parameters	3,155,328
Layers	8
Width	$d_{\text{model}} = 384$
FFN width	$d_{\text{ff}} = 1024$
Attention heads	6
KV heads	2
Attention schedule	every 4th block; final block forced to attention
Block mix	6 Gated DeltaNet + 2 attention
Dropout	0.1
Max sequence length	1024 delayed steps
Codebooks / vocab	8 / 1027
Codec	EnCodec 24 kHz, 6.0 kbps
Chunk length	8.0 s
Batch size	24
Optimizer	fused AdamW, $\beta = (0.9, 0.95)$, weight decay 0.01
Schedule	warmup 500, cosine decay from 3×10^{-4} to 10^{-5}
Precision	bfloat16

Token interleaving for multi-codebook generation. We train with a delay-pattern objective closely aligned with the multi-codebook interleaving strategy popularized by MusicGen [2]: codebook k is shifted by k positions so that a single autoregressive step predicts one token per codebook on a staircase-shaped frontier.

3 Model

3.1 Overview

The model is a decoder-only language model over residual vector quantization (RVQ) codes. Audio is tokenized into $K = 8$ EnCodec codebooks; the model consumes delay-patterned tokens and predicts the next delayed token autoregressively. Table 1 summarizes the configuration.

3.2 Token Interface and Delay Pattern

Let raw codec tokens be $z \in \{0, \dots, V-1\}^{K \times T}$ with $K = 8$ codebooks and vocabulary size $V = 1027$, including special tokens. We apply a delay pattern Δ,

$$\tilde{z}_{k,t} = \begin{cases} z_{k,t-k}, & \text{if } 0 \le t-k < T, \\ \text{PAD}, & \text{otherwise,} \end{cases}$$

so the model predicts a staircase of codebook targets rather than all raw codebooks aligned at the same step. For our 8-second EnCodec chunks, local sample metadata confirmed an effective frame rate of approximately 75 Hz, so a 600-frame raw chunk becomes 607 delayed steps. The 1024-step context therefore covers the full 8-second training chunk without truncation.

3.3 Backbone

The backbone follows the public OLMo Hybrid pattern [6]: three recurrent blocks followed by one attention block, repeated across the network, with the final layer forced to attention. In this 8-layer instance, the schedule is

$$[\text{GDN}, \text{GDN}, \text{GDN}, \text{Attn}, \text{GDN}, \text{GDN}, \text{GDN}, \text{Attn}].$$

Each block is pre-norm with RMSNorm, mixer, residual add, RMSNorm, SwiGLU MLP, and a second residual add. RVQ embeddings are summed across codebooks at each delayed step to form a single 384-dimensional token representation; there is no learned absolute positional embedding.

3.4 Attention Blocks

Attention blocks use grouped-query attention with 6 query heads and 2 KV heads, so the attention head dimension is $384/6 = 64$. Queries and keys are normalized with per-head RMSNorm, rotary position embeddings use $\theta = 500{,}000$, and PyTorch scaled dot-product attention is used on the fast CUDA path.

3.5 Gated DeltaNet Blocks

The recurrent blocks implement a paper-aligned Gated DeltaNet-style mixer. Relative to the attention geometry, the recurrent blocks shrink the q/k head width and expand the value width:

$$d_{qk}^{\text{GDN}} = 0.75 \times 64 = 48, \qquad d_v^{\text{GDN}} = 2 \times 48 = 96.$$

For each timestep, the block forms q, k, v, and gating signals a and b, together with learnable $A_{\log}$ and dt_{bias} parameters and short depthwise causal convolutions over the projected sequences. In the plain recurrent scan actually used for stable training, the state update is

$$r_t = S_{t-1} k_t, \tag{1}$$

$$S_t = \alpha_t \odot \left(S_{t-1} - \beta_t \odot (r_t k_t^\top) \right) + \beta_t \odot (v_t k_t^\top), \tag{2}$$

$$y_t = S_t q_t, \tag{3}$$

with q_t and k_t L2-normalized and with a gated RMSNorm output projection. This mirrors the intended Gated DeltaNet structure from the released OLMo-core implementation, but in the stable A100 run it executed through the plain PyTorch fallback rather than the fused FLA kernel.

3.6 Objective

The training objective is next-token cross-entropy on delayed tokens:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{k,t} \text{CE}\left(p_\theta(\tilde{z}_{k,t+1} \mid \tilde{z}_{\le t}), \tilde{z}_{k,t+1} \right),$$

ignoring pad positions and using label smoothing of 0.1. This is an unconditional codec language model: there is no text prompt, no speaker embedding, and no semantic token stream.

Table 2: A100 operating-point search for the no-FLA training path.

Configuration	Effective batch	Outcome	Role in final result
8×4	32	stable but underutilized	initial safe A100 launch
16×2	32	stable, better utilization	intermediate tuning point
24×1	24	stable, best practical tradeoff	final reported run
32×1	32	out-of-memory / unstable	rejected

4 Data and Training Setup

4.1 Dataset

We used LJ Speech [4], a clean single-speaker English corpus frequently used in TTS research. The appeal of LJ Speech for this pilot was not scale but cleanliness: if the goal is to determine whether the backbone can learn local speech structure, a narrow acoustic domain is preferable to a noisy heterogeneous corpus.

We tokenized the corpus into non-overlapping 8-second chunks and split at the utterance level to avoid train/validation leakage across chunks from the same source waveform. The resulting tokenized dataset contained 12,624 training files and 666 validation files.

4.2 Codec and Metadata

The authoritative dataset metadata came from the generated `codec_meta.json`: EnCodec 24 kHz, 8 codebooks, codebook size 1024, 6.0 kbps bandwidth, and 8.0-second chunking. One copied run configuration retained stale fallback codec defaults unrelated to the actual run, so for this report we reconstructed the codec description from the dataset metadata and saved sample manifests rather than relying on the copied run configuration alone.

4.3 Stable A100 Run

The stable run used:

- one NVIDIA A100-SXM4-80GB GPU,
- bfloat16 mixed precision,
- true batch size 24,
- 8 dataloader workers with prefetch factor 4, pinned memory, and non-blocking transfers,
- fused AdamW,
- CUDA scaled dot-product attention flash path enabled,
- `torch.compile` disabled for the stable run, and
- the fused recurrent FLA path disabled because of backward-kernel instability on the available Triton stack.

An earlier batch-32 attempt ran out of memory. Smaller batch/accumulation combinations were also tried, but batch 24 with no accumulation was the best stable operating point found under the available budget.

Table 3: Evaluation history from the stable A100 run.

Step	Train loss	LR	Grad norm	EMA val loss	PPL
200	5.0049	1.19×10^{-4}	0.28	6.7878	886.99
400	4.5167	2.39×10^{-4}	0.85	6.1530	470.13
600	4.2326	3.00×10^{-4}	0.18	5.6474	283.54
800	4.6158	3.00×10^{-4}	0.24	5.1973	180.78
1000	4.1367	2.99×10^{-4}	0.21	4.8510	127.87
1200	4.1702	2.97×10^{-4}	0.20	4.6169	101.18
1400	4.4976	2.96×10^{-4}	0.21	4.4626	86.72
1600	4.2774	2.94×10^{-4}	0.20	4.3585	78.14
1800	4.2224	2.91×10^{-4}	0.18	4.2847	72.58

4.4 A100 Run Chronology and Operating-Point Search

The final reported A100 result was not the first A100 launch. The practical training path involved a short search over operating points under tight budget constraints and unreliable pod infrastructure. The decisive run was `ljspeech_olmo_hybrid_a100_no_fla_12k_b24a1`.

Before landing on that configuration, we explored both more conservative and more aggressive operating points. Table 2 summarizes the relevant A100 configurations used during the search.

This operating-point search matters for interpretation. The reported A100 result is not merely "an A100 run"; it is the outcome of choosing the largest stable no-FLA batch regime that fit the 80 GB card without sacrificing run survival. The result therefore reflects both architectural behavior and the systems reality of the unfused recurrent fallback.

4.5 Systems Caveat

The intended fast path for the recurrent mixer was the fused Flash Linear Attention Gated DeltaNet kernel. In practice, the available pod/software stack failed in the fused backward kernel with Triton code-generation errors. As a result, the stable run reported here combines strong CUDA kernels for attention and optimization with an unfused recurrent fallback. This matters for systems interpretation: the run demonstrates architectural viability, but not the architecture's best possible throughput on its intended kernel stack.

5 Results

5.1 Validation Trajectory

Table 3 and Figure 1 summarize the recorded evaluation history up to the best locally preserved checkpoint.

The best locally preserved checkpoint is the step-1800 model. The run likely reached or began the step-2000 boundary before the pod was torn down, but we did not recover a verified step-2000 checkpoint, so step 1800 is the final trusted result.

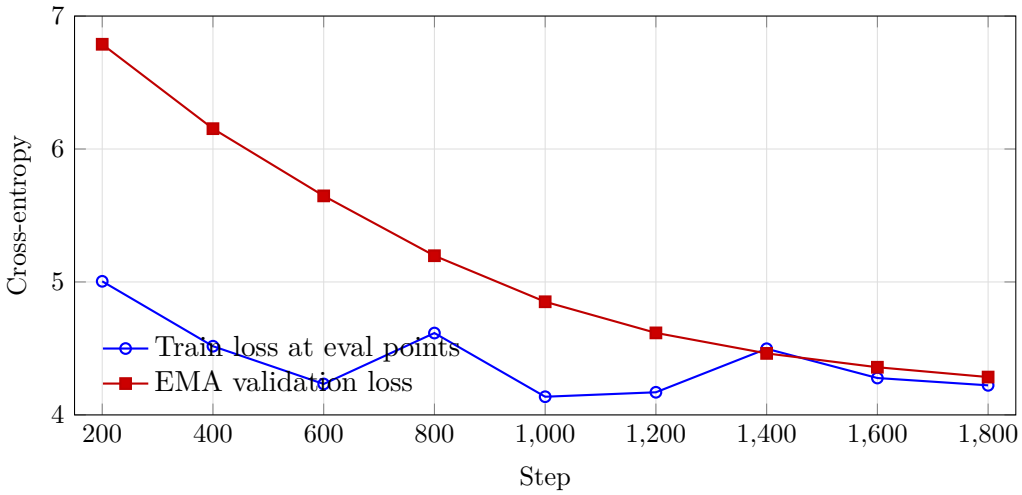

Figure 1: Training and validation loss over the stable A100 run. Validation improved monotonically through the best preserved checkpoint at step 1800.

5.2 Checkpoint Chronology on the A100 Run

The saved checkpoint cadence was every 200 steps. The locally preserved A100 checkpoints therefore form a clean chronology of the actual run:

$$200 \to 400 \to 600 \to 800 \to 1000 \to 1200 \to 1400 \to 1600 \to 1800.$$

This chronology is not incidental. It is the concrete record of the best surviving A100 run, and it is the basis for the qualitative comparison bundles used later in the report. In particular, local artifacts were explicitly pulled for checkpoints 1200, 1400, 1600, and 1800 before the pod died. The best checkpoint through the surviving run history was the step-1800 model.

5.3 Qualitative Sampling Protocol

Qualitative evaluation was performed by direct listening to locally decoded 6-second continuations from checkpoints 1200, 1400, 1600, and 1800 under a fixed prompt and consistent sampling settings. All qualitative judgments in this report are therefore author judgments rather than blinded external ratings.

Final qualitative probes used a corrected delay-space sampler rather than the earlier raw-code rollout that exaggerated timing artifacts. The improved local sampling settings were:

- one raw seed frame from a held training chunk as prompt,
- duration 6.0 seconds,
- top-$p = 0.9$,
- codebook-aware temperature schedule $0.72 \to 0.55$,
- codebook-aware top-k schedule $48 \to 24$, and
- repetition penalty 1.08 over a 48-token window.

5.4 Qualitative Outcome

The key threshold was crossed well before full convergence. The outputs were not static, hiss, or degenerate decoder noise. Instead, they exhibited:

- a clearly human speaking timbre,
- local phonetic and prosodic structure,
- partial articulation and cadence, and
- failure modes dominated by babble, weak semantics, and variable pacing.

This matters because it is a stronger result than merely reconstructing audio through the codec: the language model itself learned enough token dynamics to remain in the basin of speech-like output.

5.5 Systems Performance

Throughput on the final A100 operating point was approximately 17.9k token-equivalents per second. This was substantially faster than the local M4 experiments, but it still left the GPU underutilized relative to what the architecture should achieve with a working fused recurrent kernel. The reason is important: the bottleneck was not attention, dataloading, or basic CUDA support. It was the unfused recurrent Gated DeltaNet path. In other words, the run was strong enough to validate the architecture, but not a definitive statement about the architecture's best training efficiency.

This distinction was visible during the A100 search itself. Increasing the true batch improved memory occupancy and helped overall throughput, but did not produce the kind of saturation one would expect from a small dense transformer on the same hardware. That is consistent with a recurrence-heavy plain PyTorch fallback dominating the runtime instead of a fused recurrent kernel.

6 Discussion

The clearest positive conclusion is that the architecture was viable in this setting. Although that may sound modest, it is not a trivial result for a pilot built around a nonstandard backbone on speech codec tokens. By step 1800, the model had:

1. learned a stable speech-like token distribution,
2. improved validation monotonically up to the best preserved checkpoint,
3. produced voice-like outputs under fixed local sampling, and
4. done so without text conditioning, speaker embeddings, or a semantic token hierarchy.

That result is already enough to justify follow-up work. It suggests that the hybrid backbone is not merely compatible with speech codec language modeling in principle; it is practically viable at small scale.

At the same time, the experiment does not show that this backbone is better than a matched transformer. We do not yet have the baseline required to make that claim. What we can say is narrower:

- the backbone is viable,
- the learning dynamics on the A100 run were clean,
- speech-like audio emerged early, and
- the result is strong enough to motivate ablations, longer runs, and conditional extensions.

7 Limitations and Threats to Validity

This report has several important limitations.

No matched baseline. We did not run a same-parameter transformer baseline under the same data, tokenizer, and training budget. This prevents any claim of architectural superiority.

No human study. Qualitative evaluation was direct listening by the experimenter, not a blinded human evaluation with MOS, CMOS, or transcription-based intelligibility metrics.

Single-speaker data. LJ Speech is intentionally narrow. That is useful for isolating whether the architecture can learn speech structure, but it limits claims about general multi-speaker speech modeling.

No text conditioning. This was an unconditional codec language model. The report does not address pronunciation control, text alignment, or TTS quality.

Kernel mismatch. The stable run did not use the intended fused FLA recurrent path. The result therefore validates the model family more than it validates the final optimized training stack.

Pod volatility. The training environment was operationally fragile. Pod restarts changed SSH endpoints, local pod storage was not trustworthy, and the run likely died near the step-2000 boundary. We mitigated this by pulling checkpoints off-pod incrementally, but the final result was still shaped by infrastructure failure.

8 Conclusion

We presented a small-scale technical report on an OLMo-Hybrid-style speech codec language model trained unconditionally on LJ Speech tokenized with EnCodec 24 kHz. A 21.9M-parameter model with 6 Gated DeltaNet blocks and 2 attention blocks reached EMA validation loss 4.2847 and perplexity 72.58 by step 1800 on an A100, and direct listening confirmed clearly speech-like outputs rather than static or decoder collapse. The model remained far from semantic or production-quality speech synthesis, but it crossed the threshold that matters for a pilot: the backbone learned enough local speech structure to justify continued work.

The next steps are straightforward:

1. a matched transformer baseline,
2. a longer run beyond 1800 steps,
3. text conditioning,

4. more formal qualitative and quantitative evaluation, and

5. eventually a fully fused recurrent kernel stack.

If those follow-ups hold, this pilot will have served its purpose not as a finished speech system, but as a clean proof that this architecture family belongs in the speech-token modeling conversation.

Artifact note. The locally preserved artifacts include checkpoints at steps 1200, 1400, 1600, and 1800, the best checkpoint through 1800, and fixed-protocol sample bundles for later comparison. These artifacts were used directly in preparing this report.

References

[1] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. AudioLM: A language modeling approach to audio generation, 2022. https://arxiv.org/abs/2209.03143.

[2] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation, 2023. https://arxiv.org/abs/2306.05284.

[3] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression, 2022. https://arxiv.org/abs/2210.13438.

[4] Keith Ito and Linda Johnson. The LJ Speech dataset, 2017. https://arxiv.org/abs/1707.03982.

[5] Xiangyu Lu, Xu Wang, Haoyu Wang, Hongyun Zhou, Haiyan Zhao, Conghui Zhu, Tiejun Zhao, and Muyun Yang. DuplexMamba: Enhancing real-time speech conversations with duplex and streaming capabilities, 2025. https://arxiv.org/abs/2502.11123.

[6] William Merrill, Yanhong Li, Tyler Romero, Anej Svete, Caia Costello, Pradeep Dasigi, Dirk Groeneveld, David Heineman, Bailey Kuehl, Nathan Lambert, Chuan Li, Kyle Lo, Saumya Malik, DJ Matusz, Benjamin Minixhofer, Jacob Morrison, Luca Soldaini, Finbarr Timbers, Pete Walsh, Noah A. Smith, Hannaneh Hajishirzi, and Ashish Sabharwal. OLMo Hybrid: From theory to practice, 2026. Technical report.

[7] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers, 2023. https://arxiv.org/abs/2301.02111.

[8] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule, 2024. https://arxiv.org/abs/2412.06464.

[9] Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. SpeechTokenizer: Unified speech tokenizer for speech large language models, 2023. https://arxiv.org/abs/2308.16692.