Moonshine v2: Ergodic Streaming Encoder ASR for Latency-Critical Speech Applications

Manjunath Kudlur Evan King James Wang Pete Warden

Moonshine AI

Abstract

Latency-critical speech applications—including live transcription, voice commands, and real-time translation—demand low time-to-first-token (TTFT) and high transcription accuracy, particularly on resource-constrained edge devices. Full-attention Transformer encoders remain a strong accuracy baseline for automatic speech recognition (ASR) because every frame can directly attend to every other frame, which resolves otherwise locally ambiguous acoustics using distant lexical context. However, this global dependency incurs quadratic complexity in sequence length, inducing an inherent "encode-the-whole-utterance" latency profile. For streaming use cases, this causes TTFT to grow linearly with utterance length as the encoder must process the entire prefix before any decoder token can be emitted. To better meet the needs of on-device, streaming ASR use cases we introduce Moonshine v2, an ergodic streaming-encoder ASR model that employs sliding-window self-attention to achieve bounded, low-latency inference while preserving strong local context. Our models achieve state of the art word error rates across standard benchmarks, attaining accuracy on-par with models 4x their size while running 3x-8x faster. These results demonstrate that carefully designed local attention is competitive with the accuracy of full attention at a fraction of the size and latency cost, opening new possibilities for interactive speech interfaces on edge devices.

1. Introduction

Modern automatic speech recognition (ASR) systems are separated into two deployment paradigms: cloud-based models that leverage server-scale compute and edge models that run locally on resource-constrained devices. While cloud ASR can achieve excellent accuracy by utilizing large models and extensive computational resources, edge ASR is essential for applications where network connectivity is unreliable or unavailable, such as offline voice assistants, medical dictation in remote settings, real-time captioning for accessibility, and privacy-sensitive voice commands on mobile devices. Edge deployment also eliminates network round-trip latency and reduces privacy concerns by keeping audio data on-device.

In edge use cases, latency and transcription quality are the two key—and often competing—constraints. Achieving human-perceivable real-time performance requires minimizing time-to-first-token (TTFT) and maintaining low per-token latency, while simultaneously delivering word error rates (WERs) competitive with cloud-based alternatives. Balancing these competing objectives on devices with limited memory, compute, and power budgets remains a central challenge in practical ASR deployment.

Existing edge ASR models leverage a full-attention encoder architecture, which allows every frame to directly attend to every other frame in a sequence of speech audio. This enables powerful contextual disambiguation as it resolves locally ambiguous acoustics using distant lexical information that occurs earlier or later in a chunk of speech audio. However, full attention also introduces quadratic complexity in sequence length and imposes an inherent "encode-the-whole-utterance" latency profile: in streaming scenarios, the encoder must process the entire prefix (or wait for the complete utterance) before decoder tokens can be emitted, resulting in high TTFT that scales linearly with utterance length. In practical applications, this reduces system responsiveness and limits interactivity.

In this paper, we introduce Moonshine v2, a family of ergodic streaming encoder ASR models designed specifically for latency-critical edge applications. Moonshine v2 models employ sliding-window attention in a position-free encoder to enable low-latency streaming inference while maintaining state-of-the-art accuracy on standard benchmarks. We

. Correspondence to: Manjunath Kudlur <keveman@moonshine.ai>.

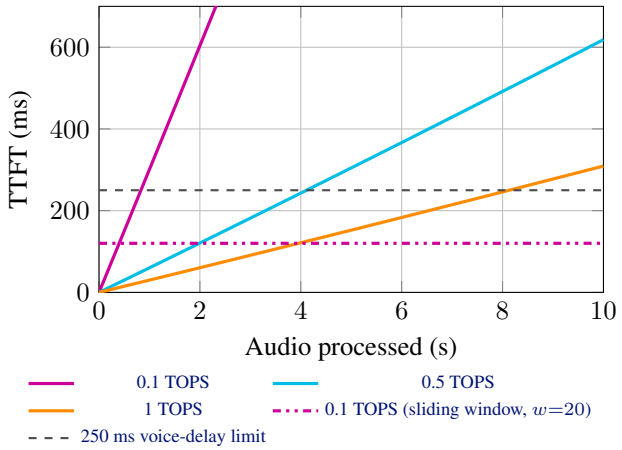

Figure 1. Illustrative time-to-first-token (TTFT) for a *full-attention* encoder as a function of audio length, for processors with different peak throughput (TOPS). The estimate includes both a linear non-attention term and a quadratic self-attention mixing term. The dotted horizontal line shows a 0.1 TOPS sliding-window encoder with $w = 20$ frames. The dashed line indicates a 250 ms one-way delay limit often used as a practical upper bound for acceptable interactive voice in private networks (Cisco Systems, 2026).

train three variants of increasing size— tiny, small, and medium—and show that the models achieve transcription quality and speed on-par with models 4x their size while running 3x-8x faster. We release the models under a permissive license, encouraging community adoption for on-device, latency-critical ASR applications.

The paper is structured as follows. Section 2 analyzes the latency-accuracy trade-offs inherent in full-attention encoders and motivates our sliding window approach. Section 3 details the Moonshine v2 architecture, including the audio preprocessor, sliding-window encoder, adapter, and decoder components. Section 4 presents our experimental setup and benchmark results across standard ASR datasets. Finally, Section 5 discusses implications and future directions for ergodic streaming ASR.

2. Motivation

This section motivates the need for low-latency, streaming-friendly encoder architectures in ASR, highlighting the trade-offs between recognition accuracy and time-to-first-token (TTFT) latency in current models.

2.1. Full-attention encoders: accurate, but latency-heavy

Many high-accuracy ASR systems rely on encoder architectures that use full self-attention over the entire input sequence. For example, Whisper (Radford et al., 2022) uses a Transformer encoder with global attention, and NVIDIA's Parakeet models build on FastConformer-style encoders (Rekesh et al., 2023).

Full attention helps accuracy because it lets each frame incorporate evidence from any other frame, enabling global disambiguation (e.g., long-range coarticulation, speaker/style consistency, and resolving locally ambiguous acoustics using distant lexical context). This ability to integrate long-range context is one reason these models achieve strong recognition accuracy. However, this same global dependency that enables superior accuracy also creates a fundamental latency bottleneck for streaming applications.

Time-to-first-token (TTFT). For latency-critical ASR, a key metric is TTFT: the wall-clock time from audio arrival to the first emitted text token. With a full-attention encoder, TTFT grows with the amount of audio that must be encoded before decoding can start. Moreover, even with a fixed model size, the attention mixing work grows quadratically with sequence length.

Figure 1 illustrates this effect for a 100M-parameter encoder processing 50 Hz features (Whisper-style). We estimate encoder compute as $\mathrm{ops}_{\mathrm{total}}(N) = 6PT + 4dLT^2$ with $T = 50N$ frames, and convert operations to time assuming a peak throughput of X TOPS (i.e., $X \cdot 10^{12}$ ops/s). The plotted curves show the resulting TTFT (ms) versus audio duration for several hardware budgets. We also include a constant TTFT line for sliding-window attention at 0.1 TOPS using $\mathrm{ops}_{\mathrm{total}}(N) = 6PT + 4dLTw$ with $w = 20$ frames (matching the Moonshine v2 streaming lookback+lookahead window).

We plot only 0.1–1 TOPS because our focus is edge deployment (phones and smaller devices), where achievable throughput is often in the 10s–100s of GOPS. A simple sanity check is *peak MAC/cycle* $\approx$ (instr/cycle)$\times$(MAC/instr), e.g., an Arm Cortex-A55 might reach $\approx$ 16 MAC/cycle; at 2.31 GHz this is $\approx$ 37 GMAC/s ($\approx$ 74 GOPS). Even when edge devices advertise multi-10s of TOPS, sustaining 1 TOPS in practice is difficult due to memory bandwidth and thermals, so we focus on the 0.1–1 TOPS regime. The horizontal line at 250 ms marks a commonly used one-way delay limit for acceptable interactive voice in private networks (Cisco Systems, 2026).

A key takeaway is that even a very strong edge-class budget of 500 GOPS (≈ 0.5 TOPS) crosses the 250 ms threshold at roughly 4.1 s of audio in this model, making "responsive" first-token latency impractical for longer utterances without streaming.

For sliding-window attention, we show only the 0.1 TOPS line because it already falls below the 250 ms voice-delay limit; higher-throughput hardware would reduce the line further.

2.2. Sliding-window attention encoders: streaming-friendly latency

A natural way to reduce TTFT is to replace full self-attention with *sliding-window* self-attention, where each frame attends only to a bounded local neighborhood. With a fixed window size w, the attention mixing cost becomes linear in sequence length ($\mathcal{O}(Tw)$ rather than $\mathcal{O}(T^2)$), and—crucially for streaming—the encoder can emit usable representations incrementally as soon as the required local context has arrived.

In a causal sliding-window encoder, the representation at time t depends only on past frames, so it can be produced immediately without waiting for future audio. If a small right context is used (lookahead), the algorithmic latency is bounded by w_{right} frames (e.g., $w_{\text{right}} \times 20\,\text{ms}$ at 50 Hz). This bounded, constant lookahead makes latency predictable and largely independent of utterance duration, enabling responsive partial hypotheses for live transcription.

3. Approach

Moonshine v2 consists of four high-level stages: an audio preprocessor, a streaming encoder, an adapter, and a decoder. We start by detailing the audio preprocessor.

3.1. Audio preprocessor

Our audio preprocessor is intentionally lightweight: it converts raw audio to a 50 Hz feature sequence (matching Whisper's feature rate) using simple operations with no right context. Many of the frontend choices were informed guesses and engineering intuition rather than a comprehensive ablation study; a full sweep over alternative frontends is cost-prohibitive for us and out of scope for this paper.

The original Moonshine model (Jeffries et al., 2024) used a full-attention encoder and a different frontend with an effective feature rate of 41.6 Hz. In Moonshine v2 we standardize on 50 Hz features to align with Whisper (Radford et al., 2022) and to simplify comparisons.

Specifically, the frontend processes audio by segmenting it into non-overlapping 80-sample windows (equivalent to 5 ms at 16 kHz), performing per-frame cepstral mean and variance normalization (CMVN) (Acero & Huang, 1995), and applying an asinh nonlinearity. The asinh function, like $\tanh$, is smooth and nearly linear around zero, but it increases logarithmically for large values rather than saturating, which we found balances compression and dynamic range effectively. Finally, two causal stride-2 convolutions reduce the frame rate by approximately a factor of four, yielding about 50 feature frames per second.

3.2. Encoder

The encoder is a standard Transformer stack with sliding-window self-attention. Each layer attends to a fixed number of past frames (left context) and, optionally, a small number of future frames (right context). We denote the attention window as $(w_{\text{left}}, w_{\text{right}})$ in frames.

No positional embeddings (ergodic encoder). We do not use any absolute or relative positional embeddings in the encoder. As a result, encoder computations are translation-invariant in time: for any local window, the same function is applied regardless of where that window occurs in the utterance. Informally, the encoder is *ergodic* in the sense that it has no explicit notion of absolute position; it can only infer structure from the content of the local context provided by sliding-window attention.

In Moonshine v2, we use $(16, 4)$ for the first two and last two encoder layers, and $(16, 0)$ for all intermediate layers. Since each encoder input frame corresponds to 20 ms of audio (50 Hz), a right window of $w_{\text{right}} = 4$ implies an algorithmic lookahead of $4 \times 20\,\text{ms} = 80\,\text{ms}$: to produce the representation at time step t for layers with lookahead, the model may use information up to frame $t + 4$, i.e., up to 80 ms of future audio.

Layers with $(16, 0)$ are strictly causal: their output at time t depends only on frames $\leq t$ (plus whatever future information has already been mixed into the current frame by earlier lookahead layers). Overall, this design keeps encoder lookahead bounded and small while still allowing limited future context near the bottom and top of the stack.

Provisional vs. finalized encoder states. We note that the right-context layers also imply that, in steady state, a *finalized* representation for time step t cannot be produced until additional future audio has arrived. In our setting, a conservative bound is 16 frames of extra audio, i.e., $16 \times 20\,\text{ms} = 320\,\text{ms}$ of future context.

For applications such as live caption display, we can still decode from *provisional* (not-yet-finalized) encoder states: the newest suffix may be less accurate, but as more audio arrives the provisional states are overwritten by finalized ones and the displayed transcription naturally improves.

3.3. Adapter

The adapter bridges the ergodic encoder and the decoder. It adds a learned positional embedding to the encoder outputs, and (when needed) applies a linear projection so that the representation dimension matches the decoder dimension. In other words, the encoder remains position-free, while the decoder receives position-aware inputs.

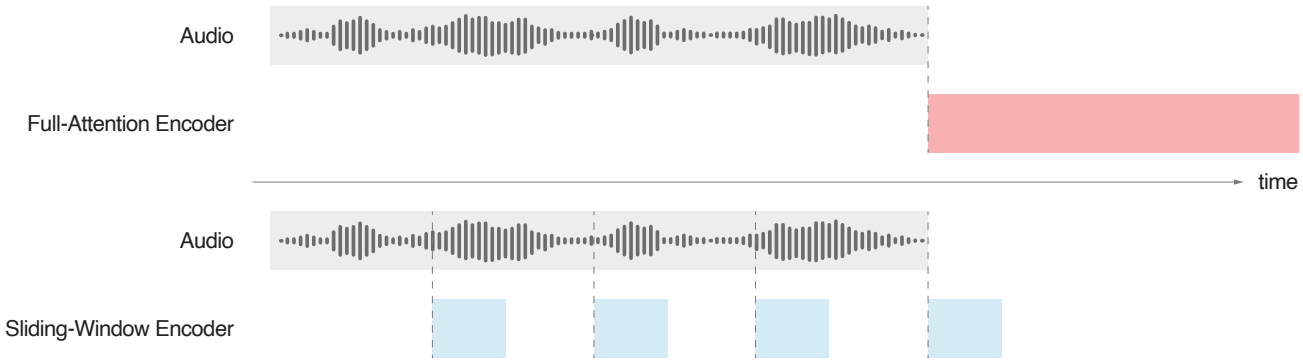

Figure 2. Conceptual TTFT timelines. In full attention, encoding begins after the entire audio has arrived. With sliding-window attention, encoding proceeds incrementally and overlaps with audio capture, so the remaining work after the last chunk is smaller.

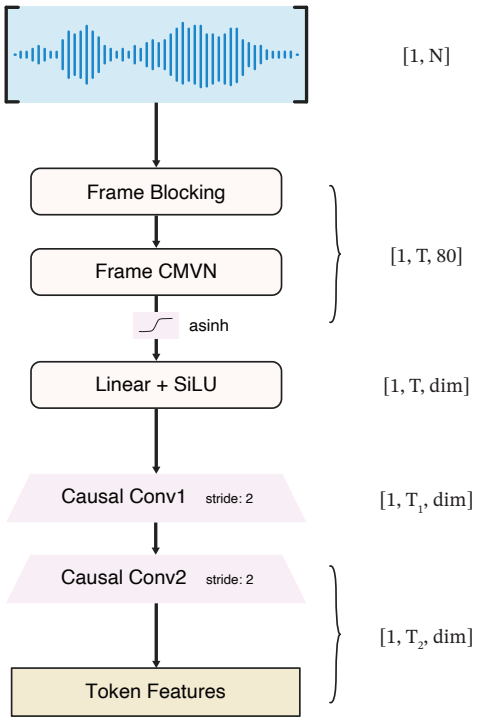

Figure 3. Audio preprocessor overview with tensor shapes, single-example. Dimensions $T = \lfloor \frac{N}{80} \rfloor$, $T_1 = \lceil \frac{T}{2} \rceil$, and $T_2 = \lceil \frac{T_1}{2} \rceil$

3.4. Decoder

The decoder is a standard causal Transformer with rotary positional embeddings (RoPE) in each layer (Su et al., 2023). It autoregressively generates text tokens and cross-attends to the adapter features.

While our ergodic streaming encoder makes the first usable features available quickly (and thus TTFT can be very low), the decoder remains autoregressive: generating a long transcript still requires a token-by-token loop, which adds latency to the full output.

A fully ergodic, infinite-streaming alternative would be to predict directly from encoder features using a linear classifier trained with CTC (Graves et al., 2006), or to use a monotonic transducer objective such as RNN-T (Graves, 2012) or Token-and-Duration Transducer (TDT) (Xu et al., 2023). Parakeet-class models follow this general direction (CTC/RNN-T/TDT-style training and decoding) and shift much of the modeling capacity into a larger encoder (Rekesh et al., 2023). Marrying these objectives with our position-free (ergodic) encoder is a promising direction that we leave to future work.

4. Evaluation & Results

We trained three model sizes (Table 1) and evaluate them on standard ASR benchmarks and latency-sensitive streaming scenarios.

Note on parameter distribution. The decoder has substantially more parameters than the encoder, largely because each decoder layer includes additional cross-attention projection matrices (in addition to self-attention), and because our decoder uses SwiGLU feed-forward blocks while the encoder does not.

4.1. Experimental setup

Training data. We use the same data sources and preprocessing pipeline as in the original Moonshine work (Jeffries et al., 2024) (see their Section 3.2, *Training data collection & preprocessing*). Relative to that setup (≈200K hours total), we add an additional 100K hours of internally prepared speech data, for a total of roughly 300K hours.

Tokenizer and optimization. We use the same tokenizer as in the original Moonshine work (Jeffries et al., 2024). We

Model	Architecture			Params (M)				
	Enc dim	Dec dim	Layers (Enc/Dec)	Pre	Enc	Adap	Dec	Total
Tiny	320	320	6/6	2.08	7.39	1.31	22.80	33.57
Small	620	512	10/10	7.74	43.49	2.86	69.27	123.36
Medium	768	640	14/14	11.86	93.66	3.64	135.77	244.93

Table 1. Moonshine v2 model architecture sizes and parameter breakdown by block.

also use the same optimizer (Schedule-Free; (Defazio et al., 2024)) with a starting learning rate of 2×10^{-3}. Training was run for 400K steps with an effective batch size of 512 on a cluster of 8 NVIDIA H100 GPUs.

Implementation. We evaluate accuracy using the implementation in the Transformers library (Wolf et al., 2019). Note that this code path does not yet perform fully efficient streaming; it relies on the flash-attention backend's sliding-window attention when available. We also measure time to first token (TTFT) using a standalone implementation using the ONNX runtime.

Benchmarks. We evaluate the accuracy of Moonshine v2 variants against similarly-sized models on the Open ASR leaderboard (Srivastav et al., 2025), reporting the word error rate (WER). We also perform empirical latency evaluations against the original Moonshine models (Jeffries et al., 2024) and the OpenAI Whisper models (Radford et al., 2022). We run these evaluations on an Apple MacBook Pro M3.

4.2. Results

Table 2 reports WERs for individual datasets. We include it for completeness, but the more informative view is the accuracy–parameter tradeoff in Figure 4. That plot shows Pareto frontiers in parameter count versus accuracy. The NVIDIA and Moonshine models lie on a similar frontier and sit above OpenAI's. Moonshine fills the lower end of the frontier (in parameter count), which is precisely the region we target: efficient ASR models for 0.1–1 TOPs and memory-constrained edge processors (e.g., sub-1 GB). NVIDIA's models, by contrast, are optimized for GPUs with tens of GB of memory and multi-PFLOP compute.

Table 3 reports TTFT, parameter count, and average WER on Open ASR benchmarks for Moonshine v2 and Whisper models. Note that while Whisper models employ full attention in the encoder, they have a fixed TTFT since all inputs are padded to 30 seconds before encoding. This table shows that Moonshine v2 models attain significantly lower fixed TTFT latency than Whisper models while achieving WER on par with significantly larger models. Moonshine v2 Medium, for instance, is 6x smaller than Whisper Large v3 but achieves effectively the same English transcription accuracy with 16x faster encoding.

Dataset	Tiny (34M)	Small (123M)	Med. (245M)
AMI	19.03	12.54	10.68
Earnings-22	20.27	13.53	11.90
GigaSpeech	13.90	10.41	9.46
Libri (clean)	4.49	2.49	2.08
Libri (other)	12.09	6.78	5.00
SPGISpeech	6.16	3.19	2.58
TED-LIUM	6.12	3.77	2.99
VoxPopuli	14.02	9.98	8.54
Average	**12.01**	**7.84**	**6.65**

Table 2. WER (%) for Moonshine v2 on Open ASR benchmarks.

Model	Params (M)	TTFT (ms)	WER (%)
Moonshine v2 Tiny	34	13.5	12.0
Whisper Tiny	39	44.0	12.8
Whisper Base	74	107.9	10.3
Moonshine v2 Small	123	65.1	7.8
Whisper Small	244	401.7	8.6
Moonshine v2 Medium	245	129.8	6.7
Whisper Medium	769	1136.3	8.1
Whisper Large v3 Turbo	809	2185.9	7.8
Whisper Large v3	1550	2173.0	7.2

Table 3. Time to first token (TTFT) at 1 second of audio and average WER on Open ASR benchmarks for Moonshine v2 and Whisper models. Sliding window attention in the Moonshine v2 encoder provides superior TTFT while achieving WER on par with significantly larger Whisper models.

Figure 5 empirically establishes the differences in TTFT between full attention and sliding-window attention encoders by comparing the original Moonshine models with Moonshine v2. For longer utterances, even the largest Moonshine v2 achieves superior TTFT to the smaller Moonshine v1 models.

5. Discussion & Conclusion

While our ergodic streaming encoder enables bounded, low-latency TTFT, Moonshine v2 still employs a full-attention autoregressive decoder. This means that once the encoder begins emitting features, the decoder must generate tokens

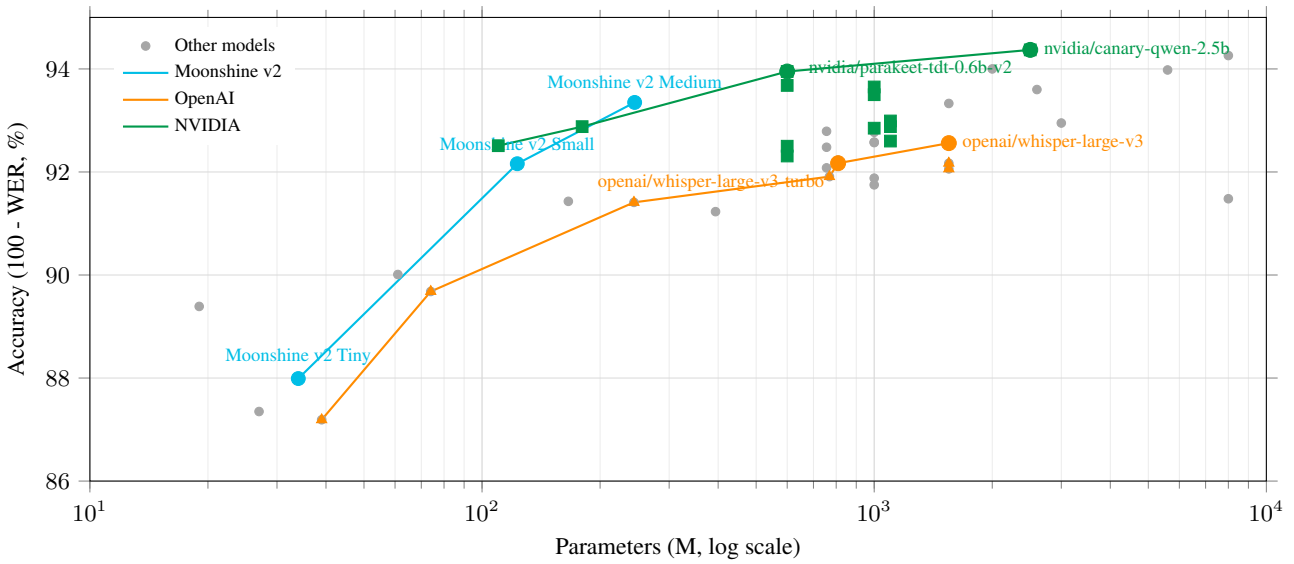

Figure 4. Accuracy vs. parameter count on Open ASR leaderboard averages.

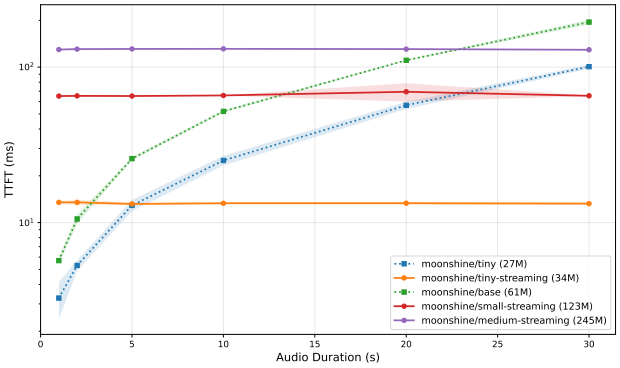

Figure 5. Time to first token (TTFT) versus input audio duration for Moonshine and Moonshine v2 models. The original Moonshine models use a full-attention encoder, which results in TTFT latency that grows with input audio duration. Sliding window attention in the Moonshine v2 encoder results in a fixed encoding latency, regardless of audio duration.

one-by-one through a serial loop. For very long transcripts, this sequential generation can add latency to the full output, even though the first tokens appear quickly. Future work could explore monotonic alignment models or streaming-friendly decoding strategies that further reduce end-to-end latency. Additionally, our current models focus exclusively on English ASR. However, the architectural principles of ergodic streaming encoders with sliding-window attention generalize naturally to other languages. Building on our prior work with specialized, language-specific models (King et al., 2025), we plan to train Moonshine v2 variants for additional languages, enabling low-latency, on-device speech recognition across diverse linguistic contexts.

We introduced Moonshine v2, a family of streaming ASR models designed for latency-critical, on-device applications. By replacing full-attention encoders with ergodic streaming encoders that use sliding-window self-attention, we achieve bounded TTFT independent of utterance length while maintaining strong transcription accuracy. Our models achieve state-of-the-art results on standard benchmarks, matching the performance of models 4x their size while running 3x-8x faster. These results demonstrate that carefully designed local attention can rival the accuracy of global attention at a fraction of the computational cost, making real-time, interactive speech interfaces practical on resource-constrained edge devices.

References

Acero, A. and Huang, X. Augmented cepstral normalization for robust speech recognition. In *Proc. IEEE Workshop on Automatic Speech Recognition*, December 1995.

Cisco Systems. Understanding delay in packet voice networks. urlhttps://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html, 2026. States that for private voice networks 200 ms one-way delay is a reasonable goal and 250 ms is a limit. Accessed 2026-01-29.

Defazio, A., Yang, X. A., Mehta, H., Mishchenko, K., Khaled, A., and Cutkosky, A. The road less scheduled, 2024. URL https://arxiv.org/abs/2405.15682.

Graves, A. Sequence transduction with recurrent neural

networks, 2012. URL https://arxiv.org/abs/1211.3711.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pp. 369–376, 2006. doi: 10.1145/1143844.1143891.

Jeffries, N., King, E., Kudlur, M., Nicholson, G., Wang, J., and Warden, P. Moonshine: Speech recognition for live transcription and voice commands, 2024. URL https://arxiv.org/abs/2410.15608.

King, E., Sabra, A., Kudlur, M., Wang, J., and Warden, P. Flavors of moonshine: Tiny specialized asr models for edge devices. *arXiv preprint arXiv:2509.02523*, 2025.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

Rekesh, D., Koluguri, N. R., Kriman, S., Majumdar, S., Noroozi, V., Huang, H., Hrinchuk, O., Puvvada, K., Kumar, A., Balam, J., and Ginsburg, B. Fast conformer with linearly scalable attention for efficient speech recognition, 2023. URL https://arxiv.org/abs/2305.05084.

Srivastav, V., Zheng, S., Bezzam, E., Bihan, E. L., Moumen, A., and Gandhi, S. Open asr leaderboard: Towards reproducible and transparent multilingual speech recognition evaluation. *arXiv preprint arXiv:2510.06961*, 2025.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing, 2019. URL https://arxiv.org/abs/1910.03771.

Xu, H., Jia, F., Majumdar, S., Huang, H., Watanabe, S., and Ginsburg, B. Efficient sequence transduction by jointly predicting tokens and durations, 2023. URL https://arxiv.org/abs/2304.06795.