Episode Book

EPISODE #5959

What RTF Actually Tells You About Speech Models

PUBLISHED	14 October 2026 at 14:26 UTC
DURATION	26:40
LISTEN	myweirdprompts.com/episode/asr-benchmarking-real-time-factor-84998/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

Most speech recognition coverage leads with accuracy — word error rate, character error rate, how close the transcript got. But that's not the number that decides whether a system is usable. This episode digs into the computational side of ASR benchmarking: real-time factor, why an RTF above one means broken rather than merely slow, and why the mean latency on your slide describes a request that never happened. From there it moves through throughput under concurrency, the memory envelope on-device, and the three environments where speech models actually run. It closes on a question worth asking out loud: is tokens per second even a meaningful metric for speech recognition, or is it borrowed from somewhere it doesn't belong?

TRANSCRIPT



Corn

Most coverage of speech recognition benchmarking leads with accuracy. Word error rate, character error rate, how close the transcript got to the reference. We've done that ourselves. But Daniel wants to talk about the thing nobody puts in the headline.



Herman

The stopwatch.



Corn

Whether any of this actually runs fast enough to be useful, and how you'd even know. He's sent us the first of two episodes on ASR benchmarking, and this one is purely computational. How efficiently models run inference, how we measure that, and how the environment you run in changes the answer before you've changed anything else.



Herman

And accuracy waits for next time. WER, CER, all of it.



Corn

So our shape tonight: the one metric that's native to speech, the metrics that sit around it, the three places you actually deploy, and how you'd build a benchmark that survives contact with reality.



Herman

So let's start with the metric that only makes sense because audio arrives at a fixed rate.



Corn

Which is the odd thing about it. Image classification doesn't have this. You hand the model a picture, it takes as long as it takes, nobody's counting the picture's runtime the way we count audio's.



Herman

Audio has a clock built into it. Real-time factor is processing time divided by audio duration. If it takes you ten seconds of compute to transcribe thirty seconds of speech, your RTF is zero point three three. The inverse gets reported more often: RTFx is audio duration divided by processing time, so it's just one over RTF. Same number, flipped so higher means better.



Corn

And a benchmark table where higher is better reads better.



Herman

It does. The Hugging Face ASR Leaderboard reports RTFx for exactly that reason. Speedups read naturally, ten times, a hundred times, three thousand three hundred and ninety times. When a number is sitting next to a bunch of other numbers in a table, the direction of goodness matters more than anybody admits.



Corn

What does an RTF above one actually mean, though. Not in the abstract, in the room.



Herman

It means the system is falling behind the speaker. Audio arrives at a fixed rate, so a chunk of duration T has to be processed in under T seconds for the system to keep up. And it doesn't have to be under T every single time. Occasional spikes are fine, the system catches up on the next chunk. But sustained, an RTF above one means the buffer fills faster than it drains, and you get overflow, and overflow means lost audio. There's no recovering the words that fell off the end.



Corn

So RTF below one isn't a target. It's the floor of a system that still works.



Herman

For anything interactive, yes. Continuous listening, live captioning, a voice agent you're talking to. Keen Research puts it bluntly, that an RTF under one isn't a nice-to-have, it's a hard requirement.



Corn

And above one is not slow. Above one is broken.



Herman

Above one is broken for that use case. For a batch job transcribing an archive overnight, an RTF of two is completely fine, it just takes twice as long as the audio's duration. Which is the first hint that all these numbers are meaningless without naming what the system is for.



Corn

RTF tells you whether you're keeping up. It tells you nothing about when the first word appears.



Herman

And that gap is where most benchmarking goes wrong. People report a single number, usually a mean, and the mean is answering a question nobody in production asked. If I tell you a model has an average latency of twelve milliseconds, that tells you almost nothing about what it feels like to talk to.



Corn

So what does.



Herman

Latency is a family, not a number. For streaming ASR you need at least three of them. Time to first byte, which is when the first output appears at all, and that's what drives whether the thing feels responsive. Word emission latency, how tightly the partial



Herman (continued)

transcript tracks the speaker, whether the words show up roughly as they're said or in clumps a second behind. And time to final, which is when the last byte lands and the transcript is actually done.



Corn

Why does the last one matter separately.



Herman

Because of what's downstream. If you're feeding that transcript to a language model, the model can't start generating until the transcript is final. So time to final is the gate. Every millisecond there is a millisecond the downstream component is doing nothing. In a voice agent pipeline, the STT time to final is a hard input to the total response budget.



Corn

Now give me the part where the mean lies.



Herman

The mean lies constantly. Gladia's example is the cleanest one I've seen. Take a hundred requests. Ninety-nine come back in twenty milliseconds. One comes back in one thousand two hundred. The mean of that distribution is thirty-one point eight milliseconds.



Corn

A number no request experienced.



Herman

Not one. Every single request was either twenty or twelve hundred, and the average you put on the slide describes a request that never happened. And that one outlier out of a hundred isn't noise. In a voice agent, that's one conversational turn in a hundred where the agent goes silent for over a second.



Corn

Which is exactly long enough for the user to decide the thing is broken.



Herman

Gladia's observation is that users start repeating themselves or talking over the agent somewhere above five hundred milliseconds. So the design target isn't the mean, it's the ninety-ninth percentile. You size the system so the worst one in a hundred is acceptable, and the mean falls out of that however it falls out.



Corn

There's something almost funny about that. A metric that gets better the more you ignore everyone who had a normal experience.



Herman

That's the whole argument for percentiles. The mean is the average citizen having an average day. Production is decided by the person having the worst one.



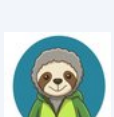
Corn

Alright, throughput. I want to hear where the tradeoff actually bites, because RTFx and latency sound like they should move together.



Herman

They move in opposite directions, and NVIDIA's own numbers show it. Parakeet, the zero point six B CTC model, on an H100, streaming. At one stream, it runs at zero point nine nine nine four nine RTFx with about twelve point four milliseconds average latency.



Corn

One stream at RTFx of one. That's basically real time and no faster.



Herman

At one stream, yes. Now take it to sixty-four concurrent streams. Throughput goes to sixty-three point seven RTFx. And per-stream latency goes to about thirty-two milliseconds.



Corn

So throughput went up sixty-four times and latency went up two and a half.



Herman

Both of those are the same model on the same silicon. Batching is the entire story. You group requests, you amortize the work across the batch, the hardware gets busy in a way it likes, and the price you pay is that every individual request now waits its turn in a queue that's longer than it was.



Corn

So "faster" is a word that needs a qualifier bolted to it every single time it's said.



Herman

Every time. If you're transcribing a million hours of call recordings overnight, the sixty-four-stream number is the one that matters, and a thirty-two millisecond per-stream latency is irrelevant because nobody's waiting. If you're running a live agent, the one-stream number is the one that matters, and the sixty-three RTFx is worthless because you're not running sixty-four conversations at once.



Corn

What about memory. Nobody puts memory in the headline either.



Herman

Memory is the constraint that kills on-device deployments, and it's the one that varies most with choices you'd think were cosmetic. Qualcomm publishes Whisper-Tiny-En numbers across two backends, both running on the NPU of a Snapdragon 8 Gen 2. Same



Herman (continued)

model, same silicon. Peak memory on the TFLITE path lands somewhere in the three to twenty-eight megabyte range. On the QNN path, twelve to fourteen.



Corn

Same weights, different runtime, and the memory envelope moves by a factor of two in one direction.



Herman

And the TFLITE range itself spans almost a factor of ten, which tells you the peak depends on how the graph gets laid out and when intermediate buffers get freed. It's not a property of the model. It's a property of the model plus the runtime plus the input shape. On a phone, that difference is the difference between fitting alongside everything else the user's running and getting killed by the OS.



Corn

Now the question Daniel actually asked, and I want a straight answer. Are tokens per second meaningful for ASR, or is that a metric we borrowed from somewhere it belongs?



Herman

There's no ASR standard for tokens per second. There isn't one, and the reason there isn't one is structural. Tokens per second is native to autoregressive language model decoding, where the output length is unbounded. You ask for five hundred words, you can ask for five thousand, and the decoder just keeps going. Throughput per forward pass is a meaningful unit there because the amount of work scales with how much you asked for.



Corn

And ASR output isn't like that.



Herman

ASR output is anchored to the input. A ten-second clip produces roughly ten seconds' worth of words, more or less, and no amount of asking changes that. The amount of output is set by the audio, not by the user. So the natural efficiency unit for ASR is audio seconds per compute second. Which is RTF.



Corn

That's a satisfying answer. The metric is shaped like the task.



Herman

It's shaped like the task. And it shows up in the leaderboard's own plans. Their future work note mentions adding token error rate, which is an accuracy metric, not a throughput metric. Nobody's adding tokens per second, because it wouldn't mean anything there.



Corn

But it does creep in somewhere.



Herman

It creeps in for LLM-decoder ASR. Architectures like Qwen3-ASR, Voxtral, Granite Speech. Those have an autoregressive decoder bolted to the acoustic encoder, and now you've imported the whole problem. The ParaASR paper puts it plainly, that the cost of autoregressive decoding scales with decoder size, which creates a fundamental tradeoff between recognition quality and serving latency.



Corn

So the decoder is the bottleneck, not the transcription.



Herman

The decoder is the bottleneck. And that's why their proposed fix is a decoding-side fix. Multi-token prediction, emitting about five verified tokens per forward step instead of one. You're not making the model better at hearing. You're making it spend fewer forward



Herman (continued)

passes saying what it heard.



Corn

Which is a trick that has nothing to do with speech.



Herman

Nothing to do with speech. It's borrowed straight from LLM inference work. Which is the whole point. For an LLM-decoder ASR system, tokens per second is a legitimate metric. It's just a property of the decoder, not a property of ASR.



Corn

So these aren't competing metrics at all. RTF, percentiles, throughput under concurrency, memory, tokens per second for the LLM-shaped systems. They're answering different questions.



Herman

And a benchmark that reports one of them is telling you a fraction of the story. The nastiest version of this is a paper that reports an RTF of zero point two and calls the model fast. Fast at keeping up. Ask it how long until the first word appears and it has no idea. Ask it what happens at a hundred concurrent users and it has no idea.



Corn

All of that assumes you can see what's happening. Now let's talk about the three places you actually run this, and how much each one lets you see.



Herman

Desktop first, because it's the clean case and it's the one everybody benchmarks on. You own the hardware, you can read utilization off the card, you can time whatever you want, you can run the same thing a thousand times and watch the distribution form.



Corn

And this is what the leaderboard does.



Herman

The ASR Leaderboard runs every model on a single A100, eighty gigabyte SXM4, driver five sixty point twenty-eight point zero three, CUDA twelve point six. Batch size sixty-four wherever memory allows, and where it doesn't, they reduce it. Forty-eight, thirty-two, sixteen, down until it fits. Every model through the same pipe so the comparison is at least internally fair.



Corn

And then they put a warning on it.



Herman

They put a warning on it. The absolute RTFx values depend on the underlying hardware and can vary substantially across systems. Everything reported there is obtained under the same setup, which is what makes the comparison valid, and it's also the limit of what it tells you. Relative comparisons transfer. Absolute numbers don't.



Corn

So the leaderboard is a ruler, not a prediction.



Herman

It's a ruler. Community numbers on desktop hardware show how far the ruler's units move. There's a project called paraketto reporting Parakeet at around eleven hundred times RTFx on LibriSpeech, eight hundred and ninety-six seconds of audio in eight hundred and eight milliseconds. An L4 benchmark from E2E Networks puts Parakeet at bfloat sixteen, batch eight, at two hundred and thirty-eight times real time.



Corn

An hour of audio in fifteen seconds.



Herman

Now, that card is a much smaller card than an A100, and it's still doing an hour in fifteen seconds. Same family of models, different silicon, and the numbers land in completely different places. Which is why the leaderboard says what it says.



Corn

Alright, Android, and I have a feeling this is where the cautionary tale lives.



Herman

This is the best piece of benchmarking writing I've read all year, and it's a blog post about a flip phone. Someone ran SenseVoice on a Z Flip 5, Snapdragon 8 Gen 2, running QNN two point four nine point zero, on six-second clips. CPU came in at seven hundred sixty-one point six eight milliseconds. The NPU, the HTP, came in at eighty-nine point seven nine.



Corn

That's a speedup of about eight and a half.



Herman

Eight point four eight. RTF of zero point one two six nine on CPU versus zero point zero one five zero on the NPU. Six-second clip, eighty-nine milliseconds. That's a good number.



Corn

But that's not the story.



Herman

That's not the story. The story is what he caught on the way there. His first pass showed a forty-five point six times NPU speedup. He looked at it and didn't believe it, and went digging, and found the CPU baseline had been running inside Android's moderate cpuset, which pins you to cores zero through two. When he moved the process into the top-app cpuset, cores zero through seven, the same CPU inference went from about four thousand eight hundred twenty milliseconds to six hundred forty-six point seven two.



Corn

Say that swing again.



Herman

Four thousand eight hundred twenty down to six hundred forty-six. Seven and a half times, from scheduling alone. Same model, same device, same code path. The only thing that changed is which cores the operating system let the process touch.



Corn

So the forty-five times speedup was measuring the OS.



Herman

It was measuring the OS scheduler, and he threw it out. And that's the part I want every listener to sit with. The published number he discarded wasn't wrong in the sense of being miscalculated. It was real arithmetic. It just wasn't measuring the thing it claimed to measure. Any Android ASR benchmark that doesn't pin cpuset is measuring the operating system, not the model, and it will be confident about it.



Corn

What about the gap between raw inference and end-to-end.



Herman

The same study makes it concrete, which I appreciate. Graph inference only, the model itself doing its forward pass, was six hundred sixty-three point two two milliseconds on CPU and thirty-six point seven five on the NPU. That's eighteen point zero five times.



Corn

Eighteen, not eight and a half.



Herman

The eight and a half is WAV in RAM to final text. And the reason the speedup gets cut in half is that the preprocessing and the decode don't run on the NPU. They run on the CPU, regardless of how fast the model is. Preprocessing was fifty-four to sixty-six milliseconds and it's sitting there on a component the accelerator can't touch.



Corn

So you can double the model's speed and get nowhere.



Herman

His line is the one worth keeping. Even if model inference is fast, if preprocessing or post-processing is heavy, the perceived speed won't improve much. You accelerate the part that was already fast and leave the bottleneck exactly where it was. The NPU was never the problem. The sixty milliseconds of resampling in front of it was.



Corn

Cloud, then, where you can't see anything.



Herman

Where you can't see anything, and you don't control the load. There's a London residential benchmark of six STT APIs from earlier this year. Groq's whisper large v3 turbo came in fastest at six hundred thirty-seven milliseconds median end-to-end. Deepgram nova-3 at seven hundred ten, but with a range from five hundred fifty-seven to fifteen hundred



Herman (continued)

fifty-one.



Corn

A three-to-one spread on the same service.



Herman

Same service, same audio, and the ninth decile is three times the median. OpenAI's whisper-1 came in at fifteen oh seven. Gemini Flash at eleven oh one, Together at eight thirty-five. All measured from a residential connection in London, which means every one of those numbers has network conditions baked into it.



Corn

So what's actually inside the budget.



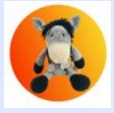
Herman

Gladia decomposes it, and the decomposition is the interesting part. Network round trip, fifty to three hundred milliseconds. Audio buffering before anything can start, about two hundred fifty. Model processing, one hundred to three hundred. And VAD endpointing, two hundred to five hundred.



Corn

The model is the smallest term.



Herman

The model is often the smallest term. Sit with that. A lot of the engineering energy in this field goes into making the model faster, and the model is one of four line items, and it's regularly the shortest one. The two biggest are things that don't feel like ASR work at all. Network and endpointing.



Corn

Endpointing deserves its own beat. What is it actually doing.



Herman

Endpointing is the system deciding the speaker has stopped talking and the utterance is complete. It has to hear enough silence to be confident, and that silence is dead time you've added by design. Keen Research separates chunk processing latency, which is what drives RTF and the partial transcripts, from final-result latency, which adds endpointing plus post-processing. Their finding is that endpointing is often the single biggest contributor to perceived latency.



Corn

Post-processing being what.



Herman

Confidence rescoring, word timestamps, pronunciation scoring if you're doing that. All the things that get attached to a final transcript but not to a partial. A partial doesn't need a timestamp on every word. A final one might.



Corn

And vendor numbers.



Herman

Vendor numbers are measured under conditions the vendor controls. Clean audio, single stream, same region, no contention. Gladia's own comparison table flags every competitor figure as not independently verified, which I respect, and notes that clean-audio benchmarks are a lower bound on latency, not a production estimate.



Corn

They're telling you their own table is soft.



Herman

They are. And that's more honest than most. The same caveat applies to their own numbers, and they say so.



Corn

So how would you actually build a benchmark that survives this.



Herman

Control the variables. Same audio, same hardware, same runtime, same batch size, same cpuset, same thermal state. Drop the warmup runs, because the first few passes are doing lazy graph construction and memory allocation that never happens again.



Corn

And report percentiles.



Herman

Report percentiles, not means. Say what your ninety-ninth percentile is, because that's the number people will experience. If you only report the mean, you've told the audience about a request that never happened.



Corn

I'd add one more. Say what you didn't control.



Herman

That's the honest version of the leaderboard warning. Every benchmark should carry a sentence saying which of its variables were fixed and which weren't, because that sentence is what tells the reader whether they can compare your number to anything.



Corn

What tools exist for this.



Herman

NVIDIA ships Riva clients. There's a streaming client with a simulate-realtime flag that feeds audio at wall-clock rate so you can measure the streaming path, and an offline client for throughput measurement. Pipecat has an STT benchmark that measures time to final segment using VAD-based speech-end detection, and its medians are useful. Nemotron 3 ASR at two hundred twenty-one milliseconds, ninety-ninth percentile two hundred fifty-two. Deepgram nova-3 at two hundred forty-seven, ninety-ninth at three hundred twenty-six. AWS at over a second, ninety-ninth percentile close to nineteen hundred.



Corn

And the accuracy versus speed frontier.



Herman

Architecture changes the answer more than size does. The leaderboard's finding is that Conformer encoders with transformer or LLM decoders win on word error rate and are slow. CTC and TDT decoders give up a little accuracy for an enormous amount of speed. Parakeet TDT zero point six B v2 sits at sixth point zero five percent WER, tenth best on the board, and three thousand three hundred ninety RTFx. FastConformer CTC Large tops the speed table at six thousand four hundred RTFx.



Corn

So the best transcribers and the fastest transcribers are different models.



Herman

Different models. And if you restrict to transformer and LLM decoders, Conformer-based systems average seven hundred fifty-eight RTFx against two hundred one for the Whisper-based ones. Three point seven seven times faster on average, before you've tuned anything.



Corn

Which means the frontier isn't a size curve. It's a set of architecture choices.



Herman

It's an architecture choice, and choosing it is a design decision you can't benchmark your way out of.



Hilbert

Your denominator.



Hilbert

You've spent an hour using the word RTF and not once have you said what audio duration you're dividing by. Nobody does. That's the problem.



Corn

Go on.



Hilbert

Is it the duration of the file, or the duration of the speech inside the file? Those are different numbers. A file with a minute of audio and thirty seconds of speech, you divide by sixty and you get one answer, you divide by thirty and you get another. Published RTF numbers almost never say which one they used. So you can't check them. You can't reproduce them. They're unfalsifiable, and everybody keeps citing them anyway.



Corn

And the cpuset thing.



Hilbert

The cpuset thing is correct and you buried it. You put it in the middle. It should have been the lede, because it's a number that changed by a factor of seven and every published Android benchmark has the same hole in it. But the cpuset at least gets reported sometimes. The denominator never does. That's worse.



Corn

How did you notice this.



Hilbert

I was asked to review a paper. Years ago. I sent in my comments, I got no response, no acknowledgement, nothing. So I kept going. Figured if nobody was going to answer me I'd just keep doing it on my own time.



Corn

You kept reviewing uncommissioned papers.



Hilbert

It's a hobby. Like birdwatching, but for denominators. I recalculated one benchmark by hand last month and got a number forty percent off what they published. Forty percent. I still have the notebook.



Corn

The notebook.



Hilbert

Physical. Legal pad. Column for the paper, column for the audio duration they used, column for the duration I could actually verify from their data, column for the recalculated number. Going back years. I've never met any of the authors. I've got no connection to the work at all.



Corn

How many of these are in there.



Hilbert

Four hundred and some. And then there's the second one.



Herman

The second notebook.



Hilbert

For suspicious batch sizes. When somebody reports a batch number that doesn't match the memory they'd need for it. The two are cross-referenced. Index at the back of each, keyed to the other.



Corn

Because of course they are.



Hilbert

If you're going to do it, do it properly. The reviewing, I mean. Not the papers. The papers are the problem.



Hilbert

I'll say one thing for the field though. The people writing the leaderboard paper were honest about the hardware caveat. Most aren't. Most put a single number in a table and let it stand there looking like a fact.



Corn

We'll take that.



Corn

Which leaves us with a question we can't really answer. If absolute RTFx doesn't transfer across systems, and cloud vendors measure under conditions they control, what would a fair cross-environment comparison even look like. Is the honest answer that there isn't one, and all we've got is relative comparisons inside a fixed setup.



Herman

I think that's close to it. You can compare two models on one machine, or two machines running one model. Comparing three numbers from three setups is theatre.



Corn

And the second question, the one that nags at me. As more ASR moves to LLM decoders, does the field drift toward LLM-style benchmarking. Tokens per second, time to first token, and away from the audio-anchored intuition that RTF encodes. Where the measurement is shaped like the input rather than the output.



Herman

That's the thing to watch. If the decoder becomes the bottleneck, the metrics follow the decoder.



Corn

The other half of this picture is accuracy. Word error rate, character error rate, and the qualitative side that neither number captures. That's next time.



Corn

If you enjoyed this one, try episode two, Local STT For AMD GPU Owners; and episode three, Safetensors or something else. This has been My Weird Prompts. Producer Hilbert Flumingtop, who has the notebooks to prove it.



Herman

Sending a prompt is how this whole thing works. Find us on Telegram at t dot me slash MWP listener bot.



Corn

And if you've enjoyed this one, a review helps people find the show.



Herman

We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **ASR Leaderboard paper (arXiv 2510.06961v4, v4, data as of 27 March 2026) [primary]**
https://arxiv.org/html/2510.06961v4

2. **NVIDIA ASR NIM Performance docs (v26.07.0) [primary]**
https://docs.nvidia.com/nim/speech/26.07.0/reference/performances/asr/performance.html

3. **Keen Research, Evaluating ASR Systems, Part 3: Latency and Responsiveness (2026-04-23)**
https://keenresearch.com/blog/evaluation/evaluating-asr-systems-part-3-latency-responsiveness.html

4. **Gladia, Latency benchmarks for streaming speech-to-text (TTLB and P99) (2026-08-28)**
https://www.gladia.io/blog/latency-benchmarks-for-streaming-speech-to-text-ttlb-and-p99

5. **Zenn, Running Local Voice AI on an Android Smartphone's NPU (2026-09-09)**
https://zenn.dev/ytmyou/articles/b9ef607f7f3c9e?locale=en

6. **charlesnchr/stt-benchmark (created 2026-02-15)**
https://github.com/charlesnchr/stt-benchmark

7. **pipecat-ai/stt-benchmark (133 stars; scoring update Sept 2026)**
https://github.com/pipecat-ai/stt-benchmark/

8. **FunASR RTF reproducibility docs**
https://github.com/modelscope/FunASR/blob/main/docs/benchmark/rtf_reproducibility.md

9. **Hugging Face Audio Course, Evaluation metrics for ASR**
https://huggingface.co/learn/audio-course/en/chapter5/evaluation

10. **ParaASR (arXiv 2607.29279v1, 2026-07-31)**
https://arxiv.org/abs/2607.29279v1

11. **Qualcomm Whisper-Tiny-En model card (Galaxy S23 / Snapdragon 8 Gen 2)**
https://huggingface.co/qualcomm/Whisper-Tiny-En

12. **E2E Networks, Benchmarking Open ASR Models on NVIDIA L4**
https://www.e2enetworks.com/blog/benchmarking-asr-models-nvidia-l4-parakeet-whisper-nemotron

13. **LokalOptima/paraketto**
https://github.com/LokalOptima/paraketto

14. **Microsoft Azure embedded speech performance evaluations (RTF definition)**
https://learn.microsoft.com/en-us/azure/ai-services/speech-service/embedded-speech-performance-evaluations