Episode Book

EPISODE #6380

Choosing the Right Voice Activity Detector

PUBLISHED	31 October 2026 at 14:59 UTC
DURATION	32:08
LISTEN	myweirdprompts.com/episode/voice-activity-detection-models-03501/
HOSTS	Corn, Herman, Vex, Neville
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	1 November 2026 at 07:41 UTC

SYNOPSIS

Voice activity detectors look interchangeable until deployment constraints reveal their hidden costs. Comparing Silero, TEN VAD, WebRTC, and Cobra shows that download size is only a fraction of the story: runtimes, licensing restrictions, and transition delays drastically alter performance. From WebRTC's lightweight Gaussian mixture models to Cobra's embedded runtime and Silero's multi-domain neural approach, choosing an effective gate depends on balancing hardware limits, background noise rejection, and conversational turn latency.

TRANSCRIPT



Corn

Daniel wants to know which voice activity detector belongs in the thing you're actually building. Silero, TEN VAD, WebRTC, Cobra, pyannote segmentation, and MarbleNet can all get called VAD, but choosing between them means deciding what lives in a browser, what fits on a phone, and what expects a GPU server. Every speech pipeline inherits the gate's mistakes. An excellent recognizer cannot recover the syllable you declined to send it.



Herman

The download sizes make these look like close competitors. Most are small enough that you could casually attach them to an email. The deployment requirements do not look remotely alike. A compact checkpoint can still arrive with a substantial runtime, while a slightly larger native library may already contain everything it needs.



Corn

The suitcase is small. The entourage isn't.



Herman

Then there is latency. Time spent calculating a speech probability is different from time spent deciding that someone has stopped talking. Confuse those and you can choose the fastest model, install it successfully, and still build an assistant that waits awkwardly before every answer.



Corn

Or interrupts me while I'm choosing a word. Your installations tend to favor the machine's conversational ambitions.



Herman

You've met some impatient defaults. I accept responsibility for installing them, not for their personalities.



Herman

There are several distinct jobs here. A detector guarding a wake-word pipeline on a microcontroller has a tight memory and power budget. A browser agent needs prompt speech onset for interruption, then sensible silence handling for turn completion. A server processing a meeting may need overlapping speakers separated into local speaker slots. Those requirements lead toward different models before accuracy enters the discussion.



Corn

I would also split the browser job in two. Stop the agent speaking when I start, but don't decide I'm finished merely because I pause. One control labeled sensitivity is concealing two decisions I might want set in opposite directions.



Herman

You would. Corn, I predict you'll keep the slower endpoint if it means finishing your sentence.



Corn

Correct. My sentence has seniority.



Herman

The comparison needs architecture, actual shipped size, transition behavior, license, and runtime. Architecture ranges from WebRTC's Gaussian mixtures to neural networks with rather different purposes. License ranges from permissive reuse to a paid commercial subscription. Runtime ranges from a compact native library to a complete speaker-processing stack.



Corn

Accuracy tables won't settle it for us. The developers test their own systems, on different audio, with different operating points. A detector can look excellent at rejecting environmental noise and less impressive at preserving quiet speech. Both measurements can be accurate descriptions of the same product.



Herman

No neutral head-to-head of all six appears in the available comparisons. The published rankings even reverse between vendors. We'll use those results to identify things worth testing, rather than pretending they form a shared league table. Also, ROC-AUC measures discrimination across thresholds. It doesn't tell you whether your chosen threshold clips someone's first word.



Corn

Nor whether the application waits politely after it. If a benchmark rewards identifying every little silence, a conversational product still has to decide which silences mean anything. I want the detector's evidence separate from the application's manners.



Herman

Keep that distinction in mind for TEN, especially. First, Silero earns its default position by being small, portable, and unusually uncomplicated to adopt. It's a compact convolutional and recurrent neural model. The JIT artifact is roughly two megabytes, while ONNX is about two point two two megabytes. Neither figure includes the general-purpose runtime you may need around it.



Corn

MIT licensed, without telemetry, registration keys, or built-in expiry. That removes several ways for a working microphone to become an account-management problem. It also means I can keep using a version that works rather than treating every upstream release as a household emergency.



Herman

That was one dependency conflict.



Corn

It reached the kitchen.



Herman

Silero accepts eight or sixteen kilohertz audio. Its published single-thread V five ONNX measurement takes about one hundred eighty-nine microseconds for a roughly thirty-one-millisecond chunk. That is comfortable computational headroom on the desktop processor used for the measurement. It is not a promise about every phone, and it says nothing by itself about endpoint delay.



Corn

The training breadth is interesting too. Corpora covering more than six thousand languages suggest a detector intended to generalize across speech, rather than a tiny speech recognizer waiting for familiar words. I wouldn't interpret that as equal validation for every language. I would interpret it as a sensible design ambition for a default gate.



Herman

Recent releases also reduced packaging friction. ONNX Runtime became optional, an offline ONNX sequence model was added, and torchaudio became an optional extra. The sequence model is sixteen-kilohertz only and releases the Python interpreter lock. That's useful for offline integration, but it is a different execution path from the streaming model whose speed we just discussed.



Corn

Fewer compulsory dependencies. An update with subtraction in it.



Herman

Silero's own comparison gives its current model a multi-domain ROC-AUC of zero point nine seven, against TEN at zero point nine three and WebRTC at zero point seven three. The consequence is that Silero is a strong starting point across varied audio in that evaluation. The caveat is quite specific: Silero selected and ran the evaluation.



Corn

TEN's challenge is more pointed than simply claiming a bigger accuracy number. It says a detector can be cheap to execute and still keep calling something speech after the useful speech has ended. That's the complaint I would notice in a voice interface, long before noticing a fraction of a processor core.



Herman

TEN VAD ships a C library, prebuilt for several platforms, plus an open ONNX model. The Linux library is about three hundred six kilobytes. Android's older Arm build is around three hundred seventy-three, and iOS around three hundred twenty. Those are shipped library sizes, which makes them more useful deployment clues than just counting neural weights.



Corn

Its native packaging is doing some of the engineering that ONNX portability leaves to me. I still need to check the actual architecture, audio conversion, and threading in my application. But the intended destination is visible. Someone has already thought about an Android application existing outside a notebook.



Herman

It accepts sixteen-kilohertz audio, with ten or sixteen-millisecond hops. On desktop, its reported real-time factor is roughly one to two hundredths. In plain language, processing takes a small fraction of the audio's duration. At that point, shaving more arithmetic may matter less than how confidently and quickly the detector changes state.



Corn

A short hop is an opportunity to make a decision, not a guarantee that you've made the right one. I'm suspicious of anyone who equates their frame interval with the time I'll experience.



Herman

TEN's README specifically alleges several hundred milliseconds of delay in Silero's speech-to-non-speech transitions, including missed short gaps between neighboring speech segments. That claim concerns the tested model and configuration. It doesn't establish that every Silero application must wait that long. Thresholds, recurrent state, smoothing, and endpoint logic all contribute.



Corn

TEN has a less convenient license story. GitHub lists NOASSERTION, which means its license detector hasn't given you a familiar standardized answer. That label is not itself a license. The ONNX release also changed the terms. Before shipping, read the actual grant rather than translating the word open into permission.



Herman

WebRTC offers the opposite sort of familiarity. Its classic VAD is a Gaussian Mixture Model over frequency-band energies, with online noise adaptation. No neural network. It comes from the older signal-processing tradition, carries BSD-three-clause terms, and can be remarkably compact. A libfvad WebAssembly build is about thirty kilobytes.



Corn

Thirty kilobytes. The documentation can outweigh the detector.



Herman

Its aggressiveness modes trade speech retention against rejection. That is useful, but it cannot manufacture acoustic discrimination the model doesn't have. Silero's environmental-noise evaluation gives it zero accuracy on ESC-fifty under that test's noise-only scoring. That's not zero usefulness everywhere. It is a striking warning about accepting noise as speech.



Corn

A gate can be excellent at noticing that silence ended while being poor at noticing why. A fan, a dropped pan, and a person are all interruptions to silence. For a quiet intercom, the old approach might be adequate. For opening a paid recognizer beside traffic, cheap detection could become expensive upstream.



Herman

Before the remaining models, remember that we're comparing different jobs sharing a label: embedded gating, conversational boundaries, and speaker segmentation.



Corn

That caller again. Who keeps approving his connection?

V

Vex

Garden gnomes! Every single one of them has a tiny pointy hat, and have you ever noticed that a pointy hat is just a microphone windscreen with legs? Two thousand and one, the gnome manufacturers quietly standardized the hatcone at exactly the same angle as a boundary microphone, and nobody has ever asked why.

V

Vex

That is why your voice activity detection models all argue about latency, because every gnome in every garden is a listening post and the models are just trying to keep up with the gnome chorus! I have the receipts, I have the garden catalogues from two thousand and one!

V

Vex

So look into the gnomes, or at least explain why WebRTC VAD is the only one that never mentions them! LOOK INTO IT!



Herman

WebRTC analyzes frequency-band energies from the audio supplied to it.



Corn

Cobra makes a more practical argument for small hardware: count everything you have to ship. Its proprietary neural network and custom native runtime total roughly four hundred fifty kilobytes. Comparing that against an ONNX model file alone understates Cobra's packaging advantage.



Herman

It supports native integration across phones, desktop, browsers, Raspberry Pi, and supported Cortex-M targets, without bringing PyTorch or ONNX Runtime. Its output is a continuous speech probability per frame. That leaves application policy in your hands, which I like. Would you pay for that integration advantage, Corn, or spend the engineering time on an open model?



Corn

For a commercial embedded product, possibly. If the alternative is maintaining the audio frontend, runtime build, and platform quirks ourselves, a subscription can buy actual work. For my personal dictation tool, probably not. The free tier is strictly noncommercial. Foundation is six thousand dollars annually, and Enterprise starts at thirty thousand.



Herman

The cost needs to be evaluated against the product, rather than against the price of downloading weights. Cobra reports a real-time factor of zero point zero three seven on a Raspberry Pi Zero. That is roughly three point seven percent of real-time processing capacity in that test. Small hardware is a real part of its pitch.



Corn

The microphone has acquired an annual budget.



Herman

Picovoice's noisy-speech benchmark reports Cobra detecting ninety-eight point nine percent of speech at its selected false-positive operating point. Silero gets eighty-seven point seven, WebRTC fifty. Those are striking gaps. They apply to that company's setup using clean read speech mixed with its chosen noise collection, not automatically to your kitchen or a telephone call.



Corn

Pyannote segmentation belongs in a different purchase discussion altogether. If I only need to know whether a person is speaking, asking for local speaker structure is extra machinery. If I need overlap in a meeting, those other detectors aren't secretly doing it and hiding the output.



Herman

Segmentation three point zero is a neural powerset speaker-segmentation model. It ingests ten-second chunks and predicts seven classes representing silence, individual local speakers, and speaker pairs. The local slots support up to three speakers in a chunk, with pairs representing overlap. Those slots still need linking across chunks to build full-recording diarization.



Corn

Meaning its who is local, not a name tag that survives the entire meeting. We shouldn't sell a segmentation checkpoint as a complete diarizer just because its output contains speaker distinctions.



Herman

Full diarization needs the surrounding pipeline. The model has MIT terms but gated access on Hugging Face, requiring acceptance and a token to obtain it. Its natural home is server-side speaker processing. We don't have a comparable streaming endpoint-latency figure for it, and the supplied information doesn't establish a useful checkpoint-size comparison. I won't substitute a download count for either.



Corn

Thank you. Popularity is famously light in memory.



Herman

MarbleNet is another neural option, a residual one-dimensional time-channel-separable convolutional network in the QuartzNet and MatchboxNet lineage. The multilingual checkpoint has about ninety-one thousand five hundred parameters, packaged in roughly five hundred kilobytes. Its compact architecture was designed to reduce parameter and computation requirements against earlier convolutional approaches.



Corn

Yet its supported distribution points at NVIDIA GPUs on Linux. Small enough to lose in a firmware folder, delivered through a GPU software stack. That's an ecosystem choice rather than proof the arithmetic requires a large accelerator.



Herman

It produces frame-based decisions at twenty-millisecond resolution from sixteen-kilohertz mono audio. Its training coverage lists six languages. Published ROC-AUC results range roughly from ninety-two to ninety-eight percent across varied evaluation sets. Those results suggest a capable model, but neither frame resolution nor aggregate discrimination gives us an end-of-turn latency measurement.



Corn

Its NVIDIA Open Model License permits commercial use, although that's a different set of terms from MIT. If NeMo and supported NVIDIA hardware are already in your server, the fit is obvious. If I'm starting with a browser, the small checkpoint alone doesn't make it the convenient choice.



Herman

Deployment should now decide the shortlist. Picovoice's criticism of general-purpose runtimes contains a useful point and a sales pitch. PyTorch and ONNX Runtime can bring more infrastructure than a tightly constrained device wants. That doesn't make ONNX inherently server-only. Browsers and phones can run it, but somebody has to account for the runtime, initialization, memory, and supported operators.



Corn

I would measure the first usable audio decision after launch as well as steady-state speed. A model that processes samples cheaply once warm can still make an application feel heavy when it starts. Download size, runtime initialization, and audio setup are separate bills. The marketing comparison tends to show whichever bill its product pays most gracefully.



Herman

Browser voice agent, then. You're building a general web tool, not firmware. Would you still defend the smallest detector if WebRTC repeatedly opens the recognizer for background sounds?



Corn

No. Silero through ONNX would be my default browser pick, provided the browser integration handles audio off the busy interface thread and the total runtime budget is acceptable. MIT matters for distributing it. I'd choose Cobra when minimizing the full shipped footprint and getting commercial platform support justify its fee. WebRTC would need a constrained use case.



Herman

TEN also has a WebAssembly build guide and browser demo, so it belongs in browser testing, especially if its transition behavior helps. Its native-platform emphasis doesn't exclude the web. That gives us a useful distinction: default recommendation versus every



Herman (continued)

viable implementation. Silero starts the evaluation, but TEN can absolutely earn the spot.



Corn

I'd test them with the agent speaking, too. A clean recording of me alone doesn't reproduce interruption. Speaker leakage can look like fresh speech. A detector's probability doesn't identify whose loudspeaker generated the sound, so echo handling and capture configuration remain part of the product.



Herman

Phone on-device moves TEN higher on my list. Its prebuilt Android and iOS libraries are concrete integration advantages. Its reported real-time factor is zero point zero zero five on an iPhone eight, versus about zero point zero five seven on a cheap Galaxy handset. That difference is a reminder to test the affordable device, not merely your own phone.



Corn

The cheap phone gets the expensive assumptions.



Herman

Those numbers are still comfortably below real time, but a mobile application shares power and scheduling with everything else. For a native phone product, I'd compare TEN and Cobra first. TEN requires reading and accepting its actual license terms. Cobra requires accepting its commercial economics. Silero remains reasonable if ONNX is already part of the app and the larger package is acceptable.



Corn

That's where existing infrastructure changes the arithmetic. If the application already ships a compatible runtime for another model, charging its full size against Silero again would be misleading. If VAD is the only reason you're adding it, ignoring that size would be equally misleading.



Herman

Microcontrollers tighten the decision further. Cobra has the clearest supported Cortex-M deployment story among these options. Specific chips still matter. Cortex-M support doesn't mean every device in that broad family has enough memory or compute. WebRTC's native code is the permissively licensed fallback when its discrimination is sufficient.



Corn

The thirty-kilobyte WebAssembly figure illustrates WebRTC's compactness, not a firmware memory guarantee. A microcontroller build has its own code size, working memory, and audio buffering. I wouldn't hand someone a browser artifact's size and promise their board will fit it.



Herman

Nor would I infer that MarbleNet is unsupported by physics because NVIDIA doesn't ship a microcontroller package. Supported integration is the deciding issue. Porting a small network can still mean reproducing feature extraction, checking numerical behavior, and maintaining inference code. What starts as a size optimization can become an entire project.



Corn

Especially when you install it before checking the operator support.



Herman

I check operator support now.



Corn

Encouraging tense.



Herman

For streaming on a normal server, Silero ONNX is my default. The permissive license, modest model, and broad audio coverage are a strong combination. TEN gets an explicit trial if boundary delay is hurting the interaction. I would compare the exact versions and post-processing we're shipping, not a remembered plot from a different release.



Corn

We should disentangle two user complaints there. Slow speech-to-silence detection delays the agent's reply. Slow speech-onset detection delays stopping the agent when I interrupt. TEN's highlighted criticism is about the former transition. It doesn't by itself prove better barge-in, even if the overall library is attractive for voice agents.



Herman

Fair correction. I nearly bundled those together. A turn detector may also wait on linguistic evidence after VAD goes quiet. A human saying, let me just... can leave a perfectly detectable gap while plainly holding the floor. Faster detection supplies earlier evidence. It doesn't tell the application what the speaker intended.



Corn

My own deployment test would retain the detector probabilities and the resulting events separately. If the probability drops promptly but the application waits, stop blaming the neural network. If the probability stays high through a quiet gap, look at the detector and its input. Otherwise every layer gets adjusted to compensate for a different layer's behavior.



Herman

What would you put in the test audio besides your very deliberate pauses?



Corn

Quiet openings, trailing consonants, breathy speech, a room fan, nearby conversation, and someone interrupting synthesized audio. Then real recordings from the intended microphones. I'd measure missed speech and false openings, but also the delays around transitions. Finally I'd listen to the clipped segments. Aggregate scores can hide an application repeatedly eating the same kind of word.



Herman

That also explains the vendor disagreement without assuming anyone fabricated results. Silero's aggregate evaluation favors its broad discrimination. TEN emphasizes its annotated speech boundaries and claims superior precision on its chosen set. Cobra fixes a false-positive operating point in noisy read speech. Those evaluations emphasize different benefits, and the competitors or versions included differ.



Corn

A vendor can win honestly on a task I don't have. I'd rather inspect the wrong decisions than argue over whose decimal has moral authority. The model's ability to distinguish speech from noise, its transition timing, and its application's decision to surrender the floor deserve separate columns.



Herman

Server-side diarization takes us to pyannote. Of these six, it is the pick when the requirement includes local speaker activity and overlap, but use it inside a complete diarization pipeline. Segmentation alone cannot assign consistent speakers across a full recording. That distinction matters more than whether you can technically collapse its output into speech versus silence.



Corn

Ten-second windows don't mean processing must take ten seconds, either. They're input context. Offline throughput can be fast while a straightforward streaming use would still wait for context. People keep taking a batch throughput figure and making it promise an interactive experience it never measured.



Herman

The current community pipeline processes an hour of audio in about thirty-one seconds on an H one hundred. That is a complete diarization throughput example, not the standalone segmentation model's latency. It makes sense for recordings and server jobs. It does not make pyannote my choice for a tiny always-on browser gate.



Corn

For an NVIDIA GPU server already using NeMo, MarbleNet gets the corresponding home-field pick. You already have the ecosystem its official packaging expects. The checkpoint is compact, commercial use is permitted, and the evaluations cover several



Corn (continued)

kinds of conversational audio. I'd avoid adding an entire GPU stack solely to run that little detector.



Herman

Agreed on the stack. Actually... the qualification is that supported Jetson devices broaden what server-shaped means here. NVIDIA hardware can be embedded hardware. Still, Linux and NVIDIA integration remain the official route, rather than a general native phone SDK.



Corn

That leaves a clean set of starting choices without pretending they're universal winners. Browser, Silero unless the full footprint or support deal favors Cobra. Native phone, TEN or Cobra. Constrained microcontroller, investigate Cobra's exact target support, with WebRTC as the free fallback. Ordinary streaming server, Silero, testing TEN against troublesome transitions. Speaker-aware processing, pyannote's full pipeline. Existing NeMo server, MarbleNet.



Herman

The runtime narrows the field; recordings from the actual application decide between the survivors. There is one more distinction we haven't really lived through here, though. Detecting a quiet interval faithfully and deciding to interrupt a person during it can be opposite achievements.

N

Neville

Neville Muddledorf, leading authority on conversational silence. I have practical experience. Well... I was assigned to check the intern demonstration, and I read the first half of the brief on the bus. I thought the real-time factor was the amount of silence it allowed before answering. I've written that on the front of my folder.



Herman

Real-time factor measures processing time against audio duration, Neville. The allowed silence is application policy, possibly combined with a turn detector. You can correct the folder without rerunning the demonstration.

N

Neville

Oh, that's much better, actually. The demonstration kept submitting my shopping request when I paused before the final item. I started writing down the unfinished words. Tuesday was carrots. Not because carrots are difficult, I was checking whether we had them. Uncle Hilbert keeps every appliance box, even after the appliances are gone, so I borrowed an empty box lid for the list.



Corn

You were testing whether it could wait for information you hadn't given it yet. That's more representative than reading a prepared sentence straight through.

N

Neville

Yes! Then I deliberately paused in the middle of ordinary requests. The pause test. I made a little mark when it cut me off. Herman, I think I wanted TEN to notice the silence quickly, but I didn't want the application to do anything final about it. Those sounded contradictory until just now. I may have blamed the wrong thing in my report.



Herman

They can both be correct requirements. Prompt acoustic evidence can feed a patient turn policy. Your unfinished requests are useful cases because the application has to cope with someone composing the request while speaking.

N

Neville

I'll change the report. I also need a column for when I actually meant to finish, otherwise I'm only collecting complaints. The box lid isn't very wide, though... I could write smaller. I brought it if either of you wants to see the marks.



Corn

Keep both outcomes. A system that never cuts you off can still be maddening if it waits long after every finished request. Your two columns would expose a tradeoff that one combined accuracy score could conceal.



Herman

That is where I'd spend evaluation time after choosing the deployment-compatible models. Tune endpoint policy against complete and incomplete utterances, then check whether improvements preserve speech onset. Don't optimize one conversational moment by quietly damaging another. Also keep the detector configuration with the result, so the next update can be compared meaningfully.



Corn

A handwritten box lid has surprisingly good version visibility.



Herman

Provided nobody recycles it.



Corn

For a commercial team, the equivalent is a small fixed regression set before every model or runtime upgrade. Include the captures that actually caused complaints, not just convenient clean speech. We aren't training on those examples here; we're checking that a previously working interaction hasn't become irritating again.



Herman

License checks belong at that same integration boundary. Silero's MIT and WebRTC's BSD terms are straightforward familiar categories. Pyannote's MIT weights still have an acquisition gate. NVIDIA's model terms permit commercial use but deserve their own review. Cobra's noncommercial free tier cannot silently become your production license.



Corn

TEN's NOASSERTION label deserves particular care because it supplies no permission by itself. Read the current terms for the actual assets you're shipping, including libraries and model. A fast detector with an unresolved distribution right is not ready for release. Neither is an extremely permissive checkpoint whose runtime you haven't successfully built.



Herman

One adjacent detail nearly disappeared under all the official runtime categories. There is a community Swift and CoreML port of MarbleNet for on-device iOS and macOS, called NeMoVAD-iOS. It demonstrates an alternative deployment route, rather than an NVIDIA-supported mobile product.



Corn

Which reinforces why parameters and supported packaging deserve separate descriptions. I would evaluate that port as a port, including its preprocessing and output behavior, instead of assuming the original checkpoint's reported results transferred untouched.



Herman

A useful neutral comparison would hold those integration details visible while testing the same speech, noise, and conversational boundaries on the actual target devices. As more ports appear, I'd like to see whether better packaging changes our picks more often than better neural weights.



Corn

Our thanks to producer Hilbert Flumingtop, and to Neville for covering the desk.



Herman

If you enjoyed this one, try episode three, Safetensors or something else; and episode two, Local STT For AMD GPU Owners. This has been My Weird Prompts, the human-AI collaboration podcast.



Corn

If this helped you choose what belongs beside your microphone, leave the show a review wherever you listen.



Herman

I'm Herman Poppleberry.



Corn

I'm Corn. See you tomorrow.

SOURCES

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

1. **Silero VAD repo, v6.2.3 (2026-09-23), MIT [primary]**
   https://github.com/snakers4/silero-vad

2. **Silero speed table [primary]**
   https://github.com/snakers4/silero-vad/wiki/Performance-Metrics

3. **Silero vs WebRTC vs TEN VAD ROC-AUC/accuracy [primary]**
   https://github.com/snakers4/silero-vad/wiki/Quality-Metrics

4. **TEN VAD repo, v1.0 / v1.0-ONNX (2025-07-11) [primary]**
   https://github.com/TEN-framework/ten-vad

5. **TEN VAD model card, RTF/lib-size table [primary]**
   https://huggingface.co/TEN-framework/ten-vad

6. **pyannote.audio 4.0.7 (2026-06-30), MIT [primary]**
   https://github.com/pyannote/pyannote-audio

7. **powerset segmentation model card, MIT, gated [primary]**
   https://huggingface.co/pyannote/segmentation-3.0

8. **Cobra product page, 450 KB, 98.9% TPR [primary]**
   https://picovoice.ai/products/voice/voice-activity-detection/

9. **Picovoice VAD benchmark methodology [primary]**
   https://picovoice.ai/docs/benchmark/vad/

10. **Foundation $6,000/yr, Enterprise from $30,000/yr [primary]**
    https://picovoice.ai/pricing/

11. **91.5K params, ROC-AUC table, NVIDIA Open Model License [primary]**
    https://huggingface.co/nvidia/Frame_VAD_Multilingual_MarbleNet_v2.0

12. **WebRTC VAD header, BSD-style [primary]**
    https://webrtc.googlesource.com/src/+/refs/heads/main/common_audio/vad/include/webrtc_vad.h

13. **Cobra V2.1, published 2025-09-26, updated 2026-01-28**
https://picovoice.ai/blog/voice-activity-detection-accuracy-improvement/

14. **Jia, Majumdar, Ginsburg, MarbleNet, ICASSP 2021**
https://arxiv.org/abs/2010.13886

15. **libfvad WASM ~30 KB, GMM description**
https://www.npmjs.com/package/auditok-webrtcvad

16. **CoreML MarbleNet port for iOS/macOS, created 2026-02-06**
https://github.com/Otosaku/NeMoVAD-iOS