Episode Book

EPISODE #6582

Why Text-to-Speech and Speech-to-Text Aren't Mirrors

PUBLISHED	14 November 2026 at 03:07 UTC
DURATION	27:33
LISTEN	myweirdprompts.com/episode/text-to-speech-recognition-asymmetry-10998/
HOSTS	Corn, Herman, Desmond, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	21 cited by the research (listed at the end)
BOOK GENERATED	14 November 2026 at 06:47 UTC

SYNOPSIS

Text-to-speech and speech-to-text look like mirror images, but the models underneath solve opposite mathematical problems. Synthesis expands a short string of characters into tens of thousands of continuous audio samples, inventing phoneme durations along the way. Recognition collapses a noisy waveform onto discrete words, discarding everything the ear doesn't need to distinguish meaning. This breakdown covers why WaveNet's sample-by-sample generation was so slow, why Tacotron 2 fails with twenty-four minutes of audio, and how T5Gemma and FreyaTTS are replacing hand-built front ends.

TRANSCRIPT



Corn

Twenty-four thousand decisions. That's what one second of speech costs a WaveNet, and every single one waits for the one before it. You want to know why your phone read a sentence aloud in two thousand sixteen and the room got older.



Herman

That's the perfect place to come in from, actually, because Daniel's prompt this week is all about that gap. He's looking at text-to-speech and speech-to-text, and he says they look like mirror images on the surface — one turns text into audio, the other turns audio into text — but he suspects the models underneath are doing quite different jobs. And he wants the fundamentals. How does a TTS model get from a string of characters to an actual sound wave. How does a recognizer get from a waveform to words. Where the real asymmetry lives, and why you can't just run one model in reverse to get the other.



Corn

And his suspicion is correct, which is the satisfying part. The mirror image holds for about one layer and then breaks.



Herman

It breaks immediately, honestly. Let me put the frame up front, because everything else hangs off it. TTS is a generative problem. You start with a short, discrete, fully specified input — a sentence, maybe sixty characters — and you have to expand it into tens of thousands of continuous audio samples that a human mouth could plausibly have produced. There is no single correct answer. A sentence can be said in infinitely many valid ways. ASR is the opposite shape: a long, noisy, continuous waveform that has to be collapsed onto a short sequence of discrete symbols, and in the process you throw away almost everything in the signal.



Corn

One direction expands, the other discards.



Herman

And that's not engineering taste. It falls out of the mathematics of the two directions. One of them is approximately a function — recording in, transcript out, mostly a right answer. The other is a distribution — text in, one sample drawn from an infinite space of renderings. The whole episode is really that distinction wearing different clothes. So let's earn it. TTS first.



Corn

Walk me from the characters.



Herman

A classic TTS engine is two parts, and the front end does something people don't expect: it destroys the text. First, text normalization — you have to expand every number, abbreviation, symbol into written-out words, because "1984" could be a year or a count and the model can't guess. Then grapheme-to-phoneme conversion, turning the spelling into phonetic transcription. Then the text gets chopped into prosodic units — phrases, clauses, sentences. What comes out the other end is what the literature calls a symbolic linguistic representation. Which is a much more honest description of the text than the text itself.



Corn

More like how a mouth works than how a keyboard works.



Herman

That's the right way to say it. The first thing a classic TTS system does is throw away the character string, because characters are an accident of spelling, and the mouth doesn't care about spelling. English is the worst case for this, obviously — "though," "through," "tough" — Hebrew is nearly the other extreme, and don't get me started, I'll be here all afternoon on the vowel situation.



Corn

Herman, hold the Semitic linguistics. We have an asymmetry to build.



Herman

Held. Barely. Stage two, then: the acoustic model. This maps that linguistic representation to an acoustic feature, typically a mel spectrogram — a time-frequency picture of the sound to be produced. The loss function is the interesting bit. It's typically L1 or L2, mean absolute or mean squared error, and that choice quietly constrains the whole output distribution to be Gaussian or Laplacian. And because the human voice lives roughly between three hundred and four thousand hertz, the loss is usually weighted to punish errors in that band harder — the scalar sits around half.



Corn

So the model gets extra credit for being wrong in the frequencies nobody listens to.



Herman

Right, and that's rational — errors at eight kilohertz matter far less to perception than errors at one kilohertz. But here's the line I want to put a pin in, because it's the whole asymmetry compressed into one sentence. Mel-frequency cepstrum features — the classic ASR feature, the thing recognizers ran on for decades — are documented as not suitable for speech synthesis, quote, "as it reduces too much information."



Corn

Say that again, slower.



Herman

The feature set that is good enough for recognition is not good enough for generation. Recognition only needs to preserve what distinguishes one word from another. Synthesis has to preserve everything the ear can hear. Same family of features, same spectrogram-shaped object, completely different information budget.



Corn

The recognizer keeps a fingerprint and calls it a face.



Herman

I'd not have put it that way, but I'm keeping it. Stage three: acoustic features to waveform, the vocoder. This is where WaveNet comes in, September twenty sixteen, DeepMind. Their move was to factorize the joint probability of the whole waveform as a product of conditional probabilities — each audio sample conditioned on all the samples before it, using dilated convolutions to reach back through history. Beautiful idea. Autoregressive, sample by sample. At sixteen kilohertz that's sixteen thousand sequential predictions for every second of audio.



Corn

Which is where my opening number came from. Twenty-four thousand at higher sample rates.



Herman

And the literature states the consequence with real feeling: the autoregressive nature of WaveNet makes the inference process dramatically slow. That's a direct quote, and you can hear the exhaustion in it. A model that produces world-class speech slower than a human reads it.



Corn

So nearly everything that happened in TTS after that is about parallelizing the generation without losing the fidelity.



Herman

You can read the whole arc as that one problem. Parallel WaveNet distilled an inverse autoregressive flow from a teacher WaveNet and got reported as a thousand times faster in production. WaveGlow, flow-based, faster than real time. Parallel WaveGAN swapped the distillation requirement for a multi-resolution spectral loss plus GAN training, so you stop needing an expensive teacher. HiFi-GAN in twenty nineteen nailed the combination — generative adversarial training, high fidelity, efficient. Meanwhile the acoustic model went through its own parallelization: Tacotron 2 in twenty eighteen is the archetype — an attention-based autoencoder that turns text into mel spectrograms, with a separate neural vocoder behind it.



Corn

Tacotron 2 has the data cliff I keep coming back to.



Herman

It's the best concrete anchor in the whole literature for how hungry clean TTS is. Tens of hours of audio for acceptable quality. Two hours, and the output degrades but stays intelligible. Twenty-four minutes, and it failed to produce intelligible speech at all. Twenty-four minutes of audio is a decent podcast segment. It's nothing.



Corn

It's one of our shorter sponsor reads.



Herman

Then FastSpeech, twenty nineteen, made the acoustic model non-autoregressive — a feedforward transformer with length regulation that predicts the whole mel spectrogram in one shot — and Glow-TTS in twenty twenty brought flow-based monotonic alignment search. Each of those is a different answer to the same question, and this is the question I want to sit on, because it's the signature TTS difficulty: how long does each phoneme last?



Corn

Nobody tells the model.



Herman

Nobody can. The input text has no durations in it. A pause before "however" is a choice. Stretching a vowel for emphasis is a choice. Every TTS architecture has to answer "how long" with something, and the answers are wildly various — Tacotron 2 learned it through attention, FastSpeech used explicit length regulation, Glow-TTS searched for a monotonic alignment, VoiceStar and the newer codec models use progress-monitoring position embeddings, FreyaTTS just predicts duration outright. The PALLE paper from last year puts the tradeoff in one line: autoregressive models suffer from slow generation and lack duration controllability, while non-autoregressive models lack temporal modeling and require complex designs. Pick your poison.



Corn

Anyway — before we go further, I think Desmond has something for us.

D

Desmond

Desmond's Money Tip. Read the newspaper in the shop and put it back, folded properly. Folded properly is the whole trick, because a paper left creased is a paper you have to buy.

D

Desmond

Do that every day for a year and you save around two hundred and forty pounds, assuming the paper is eighty pence, which it is not any more, so call it three hundred. Over a lifetime that is a small used car you will never own.

D

Desmond

I have not bought a newspaper since two thousand and nine. I read the obituaries first, to check I am not in them, and then the money pages, to check I cannot afford anything in them. Money is mostly a long wait for a small number.



Corn

Twenty-four minutes of free audio in every newspaper stand, if you fold it right.



Herman

I refuse to engage with that, and also I'm folding the Journal properly tonight. Back to duration, though — the modern era is worth one more beat, because it's where the front end starts disappearing. The current paradigm is neural codec language models. You take speech, compress it into discrete tokens with a codec, and run a language model over the tokens. Text tokens in, speech tokens out. T5Gemma, the technical report from this spring, is a four-billion-parameter encoder-decoder trained on a hundred and seventy thousand hours of multilingual speech, and it processes text directly at the subword level — no phoneme conversion at all. The four-billion-parameter model learned grapheme to sound on its own, because at that scale it can.



Corn

So the thing you said the front end exists to do — destroy the character string — the big model does internally, where we can't watch.



Herman

And where nobody had to hand-write the rules. FreyaTTS is the compact version of the same argument — a hundred and eighty-three million parameters, flow-matching diffusion transformer, and it's explicitly tokenizer-free: ninety-two symbols of Turkish character vocabulary, no phonemizer, no grapheme-to-phoneme front end, no discrete speech tokenizer. It runs at a real-time factor of about zero point one on a forty ninety, and it beats systems nearly twice its size on its own benchmark.



Corn

A model a fraction of the size, doing the linguistics itself.



Herman

The trend line is clear enough that I'll commit to it: the hand-built front end is being eaten by scale in both directions. And the duration problem didn't go away — T5Gemma's stated motivation for its position embeddings is that decoder-only architectures treat input text as a prefix competing with the growing audio sequence for positional capacity, weakening text conditioning over long utterances. Even at a hundred and seventy thousand hours, the model still has to know how long things are.



Corn

Okay. That's one direction. Flip it.



Herman

Flip it, and the first thing that disappears is the problem TTS just spent all its effort on. ASR never decides duration. The waveform already has durations in it — they're sitting right there in the signal. What ASR has instead is an alignment problem, and the literature states it as bluntly as anything: the difficulty of training comes from there being many more observations than labels. In speech audio, multiple time slices correspond to a single phoneme. You've got hundreds of frames of audio and maybe thirty characters of output, and nobody has drawn lines between them.



Corn

And the classic answer is CTC.



Herman

Connectionist temporal classification, Graves and colleagues, two thousand six. It's not an architecture — it's a loss function and an output representation for sequence labeling where input and output aren't aligned in time. The network emits a probability distribution at every time step over the label vocabulary plus one extra symbol, the blank. And here's the elegant move: label sequences are considered equivalent if they differ only in alignment, ignoring blanks. So the model can babble "C-A-A-blank-A-T" and "blank-C-A-T-blank" and both count as "CAT." An efficient forward-backward algorithm scores all the equivalent paths at once, and those scores feed back-propagation.



Corn

So CTC never learns where one phoneme ends and the next begins.



Herman

It doesn't attempt to. It marginalizes over every possible placement. And notice what that means for duration: TTS has to decide how long a phoneme lasts. ASR discovers how long it lasted, by summing over every way the audio could have been divided. Same variable, opposite status. One is a choice, one is an observation.



Corn

There's a catch in there, though. The blank trick only works because of a constraint.



Herman

Monotonicity — and this is the beautiful part for speech. CTC is limited to monotonic alignment, which would be a real problem for translation, where word order moves around. But speech is a left-to-right process. You say the beginning of the sentence before the end. For recognition, the constraint costs nothing. It's free structure.



Corn

Whereas the TTS attention has to learn the same monotonicity from scratch, and fails at it on long utterances.



Herman

Which is exactly why ARCHI-TTS, a TTS paper from this year, names the difficulty of text-speech alignment modeling as one of its two key challenges — and, this is the detail I love, it borrows an auxiliary CTC loss for its condition encoder. The ASR loss function showing up inside a TTS system, doing a job TTS couldn't do for itself.



Corn

The two fields are quietly robbing each other.



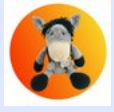
Herman

Now, CTC isn't the only answer. Attention-based encoder-decoder models avoid CTC's assumption that each output is conditionally independent of the others — they use the history of what they've already emitted. But the joint CTC-attention paper, Kim, Hori and Watanabe, documents the opposite failure mode: attention performs poorly in noisy conditions and is hard to learn early with long input sequences, because it's, quote, too flexible to predict proper alignments, without left-to-right constraints. Their hybrid — both objectives trained together — gave relative character error improvements of five to fifteen percent on Wall Street Journal and CHiME-4. Each one covers the other's weakness: CTC is rigid but stable, attention is expressive but can wander.



Corn

CTC assumes independence, attention allows time travel, and the hybrid forces monotonic alignments while dropping the independence assumption.



Herman

You've read that paper, I can tell. And ASR has had its own speed race — CASS-NAT swaps word embeddings for token-level acoustic embeddings extracted using CTC-derived boundaries, generating all tokens in parallel, roughly a twenty-four times inference speedup at close to the same error rate. Same story as TTS: autoregressive quality, non-autoregressive speed, get both by hook or by crook.



Corn

Here's where I want to push, Herman, because there's a data point that's stuck with me since you mentioned Tacotron's twenty-four minutes. The German segmentation work.



Herman

The nicest demonstration of the asymmetry in kind, not just in size. A twenty-twenty paper assembled over seventeen hundred hours of German, including unlabeled speech, by using a CTC-pretrained model to bootstrap segment alignments out of raw audio — then trained a hybrid model on all of it and got twelve point eight percent error on a benchmark where the prior baseline was fourteen point four. Seventeen hundred hours, multi-speaker, noisy, partially unlabeled, and recognition thrives on it.



Corn

Tacotron needed tens of hours of clean studio single-speaker audio and died below two.



Herman

That's the whole data asymmetry. Whisper's entire thesis is that weak supervision at scale works for recognition — huge weakly-labeled corpora, imperfect labels, and the model comes out robust. Try that with TTS and you get a voice that sounds like everyone and no one. Recognition learns to be indifferent to everything except word identity. Synthesis learns to be a specific voice, which means every imperfection in the recording becomes a feature of the output. There's an Interspeech paper from last year that makes this vivid: LLM-based zero-shot TTS tends to preserve the acoustic environment of the audio prompt. Give it a noisy reference clip and the synthesis is noisy — the model can't tell the room from the voice.



Corn

A recognizer learns to ignore noise. A generator learns to produce it.



Herman

That's the cleanest one-line version of why the directions aren't mirror images, and I want to lay the asymmetry out properly now, because it's really four separate claims that people blur into one. First, the information budget runs in opposite directions. ASR compresses thousands of samples per word down to a handful of symbols and discards pitch, timbre, accent, emotion, room acoustics — everything that isn't word identity. TTS has to invent all



Herman (continued)

of it back, from nothing, for every utterance. The mel-cepstrum line is this claim in miniature.



Corn

The fingerprint and the face again.



Herman

Second, one direction is approximately a function and the other is a distribution. For a given recording there is a right transcript, modulo homophones and disfluency. For a given sentence there is no right rendering — infinitely many, all valid, differing in pitch contour, duration, micro-timing. That's why the literatures look so different on a shelf. TTS papers are full of generative machinery — GANs, flows, diffusion, flow matching — because you're sampling from a distribution. ASR papers are full of loss functions and alignment algorithms — CTC, forward-backward, Viterbi — because you're estimating a mapping. Even the losses encode it: L1 and L2 on acoustic features for TTS, which forces Gaussian or Laplacian output distributions, versus cross-entropy over label sequences for ASR.



Corn

You can literally tell which paper is which by the verbs.



Herman

Third, duration, which we've done — decided in one direction, discovered in the other. Fourth, monotonicity, which is a gift in one direction and a burden in the other. Free structure for the recognizer, a learned skill that degrades over long utterances for the synthesizer. Which takes us to the question Daniel actually asked: why can't you just run one of these backwards?



Corn

I've been assuming it's the obvious answer — the sizes don't match. A softmax over thirty characters can't be inverted into twenty-four thousand float samples per second.



Herman

That's the first compounding reason, the cardinality problem, but it's the shallowest of them. Here's the deeper one: the discarded information is gone. Not hidden, not compressed reversibly — gone, by design and by training objective. The ASR encoder has spent its whole training life learning to not care about everything that makes a voice a voice. Run it backwards and you get something word-shaped. You don't get something voice-shaped.



Corn

Because the shape it learned is the discarding.



Herman

And the third reason is the one I find most satisfying, because it kills the idea even if you fix the first two. The losses are not inverses. CTC marginalizes over alignments to produce a likelihood of a label sequence. There is no corresponding operation that takes a label sequence and marginalizes out to a waveform distribution. The forward-backward algorithm has no generative dual in that framework. You can't un-marginalize. So the architecture is reversible — an encoder-decoder transformer runs either way, that's just wiring — but the objective is not. You cannot invert a loss function that was designed to throw information away.



Corn

The wiring runs backwards. The purpose doesn't.



Herman

Which also predicts the ways it goes wrong correctly, and this is worth checking any theory against. If the directions are truly different in kind, the ways they break should share nothing. And they don't. ASR fails by hallucination — inventing plausible text for noise, the thing Whisper is still known for — and by confusion under noise. TTS fails by unnaturalness, by duration and alignment drift on long inputs, and by reproducing the noise in its conditioning prompt. Opposite failures, exactly as predicted.



Hilbert

— it's a sample, that's what it is.



Hilbert

You two have been saying specification all afternoon, Herman especially. A rack vocoder doesn't get specifications either. I spent a season with one in a studio in Bristol — the phase reconstruction went wrong on every sustained vowel and nobody above me could hear it, so it was my job, every take, to listen for the warble. Metallic, like a coin settling in a saucepan. I can still hear it in current systems. Herman's clean models, some of them. I won't say which on the air, because you have to hear it yourself and my impression of it would just mislead you.



Herman

Do the impression.



Hilbert

No.



Hilbert

It's like… nnnnyooo. That's not it. Forget I did it. The other thing — your information budget framing undersells it, Corn. What the recognizer throws out isn't just information, it's the part of the signal that makes a voice sound like a person. I heard it leave, in real time, through a pair of NS-10s that had one blown tweeter, which the engineer refused to replace because he said the midrange was honest now. He kept a logbook. I keep a logbook now too — every TTS system I've heard since some point in the nineties, one line each, verdict only. My handwriting was better in the earlier systems. The newer the entry, the worse the writing. There are four hundred and six of them and I can find any one in under a minute. Claws don't help with the indexing, but the pages hold a crease well.



Corn

The handwriting degrades with the quality of the voice?



Hilbert

The handwriting degrades with the year. Draw your own conclusions.



Herman

The warble on sustained vowels — that's phase incoherence, and it's exactly what the generative losses were built to fix. L1 on the spectrogram never touches phase, so a system trained on it inherits whatever the phase reconstruction does. That's presumably why you hear its descendants in the new models.



Hilbert

The new models hide it in the consonants, mostly. It never fully left. And the logbook is not available for inspection, before either of you asks.



Corn

I'd have asked politely.



Hilbert

You'd have asked with the microphone on. Four hundred and six. The forty-ninth entry has a coffee ring on it and I've decided that counts as data.



Herman

Let's pull the thread back, because Hilbert's warble points at the misconception we should close on. The single most common wrong belief about this whole subject is that TTS and ASR are mirror images — that one is the other run in reverse, so a sufficiently clever engineer could flip one and get the other.



Corn

And the correction is that the architecture can be flipped and the objective can't. CTC marginalizes over alignments to get the likelihood of a label sequence. There's no operation that goes the other way — from a label sequence out to a waveform distribution. The forward-backward algorithm has no backwards that produces sound.



Herman

The wiring is symmetric. The purpose is not. Every other difference we covered today — the features, the losses, the duration question, the data requirements — falls out of that one.



Corn

Here's where my head is at, though. Both directions are converging on the same architecture: a big transformer over tokens. TTS is dropping its phonemizers, ASR never had a front end to drop, and Hilbert's four hundred and six log entries are presumably converging on something too. But text tokens are a specification and audio tokens are a sample, and no amount of convergence in architecture erases that difference in kind. Whether that convergence ever produces one shared model, running both ways with one objective —



Herman

The literature doesn't have an answer. The closest thing is a TTS model borrowing a CTC loss as a helper, which is a borrowed tool, not a reversible model.



Corn

Then we'll have to be the ones watching for it. If you've made it this far into waveform mathematics with two animals and an anteater's logbook, the review button on your podcast app is right there, and it costs you nothing, unlike a newspaper.



Herman

Producer credit to Hilbert Flumingtop, who has been at the desk this whole time and will deny that the impression was inaccurate. For more along these lines, there's episode ten, How ASR Went From Frustration To ... Whisper Magic; episode three, Safetensors or something else; and episode seven, Building Custom ASR Tools. This has been My Weird Prompts — find everything at my weird prompts dot com.



Corn

See you tomorrow, Herman.



Herman

See you tomorrow, Corn.

SOURCES

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

1. **T5Gemma-TTS Technical Report, arXiv:2604.01760v1, submitted 2026-04-02 [primary]**
   https://arxiv.org/abs/2604.01760v1

2. **FreyaTTS: A Compact Tokenizer-Free Flow-Matching Transformer for Turkish-First Speech Synthesis, arXiv:2607.09530v2, submitted 2026-07-10, revised 2026-07-22 [primary]**
   https://arxiv.org/abs/2607.09530v2

3. **ARCHI-TTS: A flow-matching-based Text-to-Speech Model with Self-supervised Semantic Aligner and Accelerated Inference, arXiv:2602.05207v1, submitted 2026-02-05, accepted ICASSP 2026 [primary]**
   https://arxiv.org/abs/2602.05207v1

4. **VoiceStar: Robust Zero-Shot Autoregressive TTS with Duration Control and Extrapolation, arXiv:2505.19462v2, submitted 2025-05-26, revised 2025-05-31 [primary]**
   https://arxiv.org/abs/2505.19462v2

5. **Pseudo-Autoregressive Neural Codec Language Models for Efficient Zero-Shot TTS (PALLE), arXiv:2504.10352v3, submitted 2025-04-14, revised 2025-08-05, accepted ACM MM 2025 [primary]**
   https://arxiv.org/abs/2504.10352v3

6. **Improving Noise Robustness of LLM-based Zero-shot TTS via Discrete Acoustic Token Denoising, arXiv:2505.13830v2, submitted 2025-05-20, revised 2025-05-22, Interspeech 2025 [primary]**
   https://arxiv.org/abs/2505.13830v2

7. **VoiceCraft-X: Unifying Multilingual, Voice-Cloning Speech Synthesis and Speech Editing, arXiv:2511.12347v1, 2025-11-15, EMNLP 2025 [primary]**
   https://arxiv.org/abs/2511.12347v1

8. **FuseCodec: Semantic-Contextual Fusion and Supervision for Neural Codecs, arXiv:2509.11425v2, submitted 2025-09-14, revised 2025-09-29 [primary]**
   https://arxiv.org/abs/2509.11425v2

9. **ESTVocoder: An Excitation-Spectral-Transformed Neural Vocoder Conditioned on Mel Spectrogram, arXiv:2411.11258v1, 2024-11-18, NCMMSC2024 [primary]**
   https://arxiv.org/abs/2411.11258v1

10. **Joint CTC-Attention based End-to-End Speech Recognition using Multi-task Learning, arXiv:1609.06773v2, submitted 2016-09-21, revised 2017-01-31 [primary]**
    https://arxiv.org/abs/1609.06773v2

11. **Audio-Visual Speech Recognition With A Hybrid CTC/Attention Architecture, arXiv:1810.00108v1, 2018-09-28, IEEE SLT 2018 [primary]**
https://arxiv.org/abs/1810.00108v1

12. **A CTC Alignment-based Non-autoregressive Transformer for End-to-end ASR (CASS-NAT), arXiv:2304.07611v1, 2023-04-15, IEEE TASLP [primary]**
https://arxiv.org/abs/2304.07611v1

13. **Online Hybrid CTC/Attention End-to-End ASR Architecture, arXiv:2307.02351v1, published IEEE/ACM TASLP vol. 28 (2020), pp. 1452 [primary]**
1465
https://arxiv.org/abs/2307.02351v1

14. **CTC-Segmentation of Large Corpora for German End-to-end Speech Recognition, arXiv:2007.09127v2, 2020-07-17, revised 2020-10-05, SPECOM 2020 [primary]**
https://arxiv.org/abs/2007.09127v2

15. **Self-Attention Networks for Connectionist Temporal Classification in Speech Recognition (SAN-CTC), arXiv:1901.10055v2, 2019-01-22, revised 2019-02-19, ICASSP 2019 [primary]**
https://arxiv.org/abs/1901.10055v2

16. **Towards Personalization of CTC Speech Recognition Models with Contextual Adapters and Adaptive Boosting, arXiv:2210.09510v3, 2022-10-18, revised 2022-11-13, SLT 2022 [primary]**
https://arxiv.org/abs/2210.09510v3

17. **Wikipedia, Speech synthesis, last edited 2026-11-09**
(retrieved 2026-11-14)
https://en.wikipedia.org/wiki/Speech_synthesis

18. **Wikipedia, Speech recognition, last edited 2026-11-13**
(retrieved 2026-11-14)
https://en.wikipedia.org/wiki/Speech_recognition

19. **Wikipedia, Whisper (speech recognition system), last edited 2026-10-29**
(speech_recognition_system) (retrieved 2026-11-14)
https://en.wikipedia.org/wiki/Whisper

20. **Wikipedia, Deep learning speech synthesis, last edited 2026-06-14**
(retrieved 2026-11-14)
https://en.wikipedia.org/wiki/Deep_learning_speech_synthesis

21. **Wikipedia, Connectionist temporal classification**
(retrieved 2026-11-14)
https://en.wikipedia.org/wiki/Connectionist_temporal_classification