Episode Book

EPISODE #6716

Choosing Open-Source Frameworks to Fine-Tune TTS

PUBLISHED	20 November 2026 at 18:19 UTC
DURATION	27:33
LISTEN	myweirdprompts.com/episode/fine-tuning-open-source-tts-60990/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
BOOK GENERATED	20 November 2026 at 21:20 UTC

SYNOPSIS

Fine-tuning text-to-speech models on a single speaker involves navigating legacy architectures and complex licensing. This guide walks through the classic open-source stack, contrasting Tacotron 2's two-stage pipeline with VITS's end-to-end flow. It examines practical realities across maintained forks like Idiap's coqui-tts, Piper's home-automation deployment, XTTS weights restrictions, and StyleTTS 2's honest hardware benchmarks.

TRANSCRIPT



Corn

Two thousand epochs. That's the number that's been stuck in my head all week, and we'll get to why. Because what Daniel's asking this time builds straight on our earlier discussion about voice cloning the hard way, fine-tuning a model on hours of one person's recorded voice. We dealt with the recordings and the frustration of training then. Now he wants the map. Which frameworks do people actually use when they're fine-tuning a text-to-speech model on hours of one person's recordings. The classic open-source routes first, then the newer generation of fine-tunable models, and what each one is like to actually sit down and use.



Herman

And the honest answer to which framework is that it depends on which era of TTS you're standing in, because the eras are different tools with different assumptions baked in.



Corn

Which is why we're going to walk the timeline.



Herman

I want to frame the real question first, though, because fine-tuning TTS on one voice is not one task. It's a stack of decisions. Acoustic model versus vocoder, train from scratch versus warm-start from a checkpoint, spectrogram-based versus codec-token-based. And the framework you pick locks in most of those decisions before you've recorded a single minute. Pick Tacotron 2 and you've committed to a two-stage pipeline. Pick VITS and you've committed to end-to-end. Pick one of the newer backbones and you've committed to someone else's pretraining, for better and worse.



Corn

So the choice is less "which tool" and more "which set of consequences."



Herman

Exactly the frame I'd use. And in practice there are two families we'll move between today. The classic open-source toolchains, meaning Tacotron 2, VITS, the Coqui stack with XTTS, Piper, Tortoise, StyleTTS 2. And then the newer wave of large pretrained models



Herman (continued)

that ship fine-tune scripts alongside them, GPT-SoVITS, F5-TTS, Orpheus, OmniVoice, Qwen3-TTS, plus the Unsloth notebooks. And I'll say up front, the axes we judge each on, because in this space they matter as much as the architecture. Data volume, GPU memory and time, licence, and maintenance status. The last commit date of a TTS repo tells you more about your next three months than the paper does.



Corn

The licence especially. A voice model is a long-term commitment, and so is the framework you build it with.



Herman

Start at the beginning, then. Tacotron 2. This is the archetypal classic route, NVIDIA's reference implementation, and architecturally it's the one that taught everyone the vocabulary. It's two-stage. The Tacotron part predicts mel spectrograms from text, and then a separate vocoder, NVIDIA's WaveGlow, which ships as a git submodule, turns those spectrograms into actual audio. Two models, trained separately, glued together.



Corn

And the fine-tuning move, the original one, is the warm start.



Herman

The warm start flag, yes. You pass a checkpoint path with the warm start flag, and it loads the pretrained weights while ignoring the text embedding layers. Which is the clever bit, if you think about what it's assuming. The mel-prediction machinery, everything the model knows about how spectrograms behave, transfers. The text embeddings, which were learned for a particular language and symbol set, get retrained for your data. It's a clean split between what generalizes and what doesn't.



Corn

The licence is BSD-3-Clause, which is permissive, so that part's friendly. What's not friendly?



Herman

The stack underneath it. It's pinned to PyTorch one point oh and Apex on CUDA. That pin is doing a lot of talking. Nobody pins to a framework that old because they're actively maintaining the repo, they pin because nobody's touched it in years. So Tacotron 2 today is less a tool you use and more a fossil you learn from. The concepts it taught, warm starts, the spectrogram plus vocoder split, those still run the whole field.



Corn

A museum piece with a gift shop.



Herman

It really is. Now VITS, which is the first real break from that two-stage world. MIT licence, one model, no separate vocoder at all. It's end-to-end, and the way it gets there is a combination of variational inference, normalizing flows, and adversarial training, with a stochastic duration predictor bolted on, which is my favorite detail. Because duration is stochastic, the same text can come out with different rhythm on two runs. You synthesize the same sentence twice and the pacing differs slightly, which is much closer to how an actual person speaks.



Corn

That's the line I'd sell it on, if selling it were legal in this state.



Herman

The catch is that the reference implementation is exactly that, a research reference. It ships trained on LJSpeech, single speaker, or VCTK for multi-speaker. Getting your own data in means a grapheme-to-phoneme conversion step, espeak for that, and compiling the monotonic alignment search from source. It's the kind of setup where the README assumes you've read the paper.



Corn

So nobody actually meets VITS directly. You meet its descendants.



Herman

Mostly through Piper, which we'll get to. VITS is the ancestor that everyone's running without knowing the name. Okay, so the practical classic route, the one people actually recommend when you ask this question on a forum, is Coqui TTS. And here's where the story gets strange, in a very open-source way. The original Coqui repository is unmaintained, because the company behind it, Coqui, shut down in January twenty twenty-four, and XTTS lives on in community forks. The one I'd point to is a maintained fork by Idiap, a Swiss research institute, published as the PyPI package coqui-tts, currently at version zero point two seven something, MPL-2.0 code, and it supports Python three point ten through three point fourteen, which tells you it's actually alive.



Corn

A maintained fork of a dead company's code. That's the classic route now.



Herman

It is, and it works. Coqui's own docs say you can get reasonable fine-tuning results with only a couple of hours of data, against roughly twenty-four hours of LJSpeech to train an English model from scratch. That's the headline tradeoff in the whole classic era. Hours versus a full day of recordings. XTTS, their flagship voice cloning model, has its own separate fine-tuning scripts in the codebase.



Corn

I'm guessing there's a licence wrinkle coming, because you said MPL for the code with particular care.



Herman

There is, and it's the instructive one. XTTS-v2 speaks seventeen languages and clones zero-shot from about six seconds of audio, with fine-tuning supported in the codebase. The code is MPL. The weights are under the Coqui Public Model License, CPML, which is a separate licence, more restrictive, and it governs the actual thing you care about, the voice you produce. People trip on this constantly. They check the code licence, see something permissive, and assume they're clear.



Corn

The code is a formality. The weights are the product.



Herman

Everything the model knows lives in the weights, and the code is just the machinery for reading them. And the irony of XTTS-v2 specifically is that it's arguably the most usable classic route for exactly this task, clone from six seconds, refine with a fine-tune, and it's the one where the dead company's licence still sits on the weights like a lid.



Corn

Piper next. This is the one that actually ships, isn't it.



Herman

Piper is the one that ships. GPL-3.0, now under the Open Home Foundation. And by ships I mean it's the TTS inside home assistant setups everywhere, the thing reading out your shopping list. Your training data is a pipe-delimited CSV of audio filename and text, about as simple as data formats get. Training is one command, and to fine-tune you pass a checkpoint path to an existing Piper voice, which speeds training up a lot, and it even works across languages, with the caveat that only the medium-quality checkpoints do it without extra fiddling on audio settings.



Corn

And the reference hardware is where my sloth ears go flat. An A6000 at forty-eight gigabytes, or a thirty-ninety at twenty-four.



Herman

That's the reference, yes, though people report getting it to run on eight-gig cards, so it's not a hard wall. The data guidance is the concrete part, and it's the two-thousand epoch number you opened with. Under about five hours of fine-tune data, the guide says expect on the order of two thousand epochs. With ten to fifteen hours, six hundred fifty to eight-fifty. Which inverts what beginners assume. Less data doesn't mean a quicker run, it means many more passes over the same material.



Corn

The model reads your five hours until it's memorized the order of your coughs.



Herman

And there's a warning in their training guide that I love, which is that the validation score, the mean opinion score the pipeline computes, is noisy and misses wrong phonemes. So you cannot pick your best checkpoint by the number. You have to listen. There's no metric that replaces your ears. The output, when you're done, is a small ONNX file that runs fast inference on a CPU, which is the whole design point. You're not shipping a research model, you're shipping something the size of a spreadsheet.



Corn

For a sloth on a machine with no GPU, that last sentence is the entire sales pitch. What's the catch?



Herman

The Open Home Foundation says it's looking for maintainers. Read that twice. The most deployed open TTS training route in the home automation world is openly hoping someone steps forward.



Corn

The classic routes so far. A fossil, an ancestor, a dead company's fork, and a volunteer vacancy.



Herman

Two more classic entries and they're the odd pair. Tortoise, Apache-2.0, and it's odd because it isn't a fine-tuning route at all, as far as its README goes. It clones at inference time from a few reference clips you drop in a folder, and fine-tuning isn't covered. And it's famously slow, the author's own figure is one medium sentence every two minutes on a K80.



Corn

Two minutes per sentence. I could say it myself faster, and I'm a sloth.



Herman

Measurably faster, yes. Tortoise mattered because it pointed at the future, inference-time cloning from a handful of clips, but as a fine-tuning route it's a nonstarter. StyleTTS 2 is the opposite, a serious training pipeline with the most honest documentation of the whole classic era. The code is MIT, but the pretrained models carry conditions, you must tell listeners the speech is synthesized unless you have the speaker's permission, and inference depends on a GPL phonemizer, so even the permissive code ships with a copyleft tail.



Corn

So the licence is three things at once, depending on which file you're holding.



Herman

And the numbers are refreshingly blunt. Fine-tuning starts from a LibriTTS checkpoint with the fine-tune script, and the default config uses about one hour of LJSpeech, roughly a thousand samples, for fifty epochs, which runs about four hours on four A100s. And the author says outright the result is slightly worse than twenty-four hours of data trained from scratch, which took about two and a half days on the same four A100s.



Corn

He published the fact that his shortcut is worse than the long way. That's the most trustworthy sentence in TTS documentation.



Herman

There's also a practical gotcha in the README, which is don't use fp16 in the first training stage, you get NaN losses. Half precision on the early phase just falls over. Small detail, saves somebody a weekend.

L

Larry

Are you tired of your voice sounding like only one voice? Then you need VoiceClone-O-Matic, the world's first framework that fine-tunes your entire personality from just two hours of humming into a microphone.

L

Larry

It warm-starts your soul, ignores your text embeddings, and ships with a separate vocoder for each of your relatives. Results not typical. Side effects may include spontaneous grapheme-to-phoneme conversion, PyTorch one point oh nostalgia, and a monotonic alignment that never quite aligns. Void where prohibited. Ask your doctor if your voice is right for you.



Herman

Two hours of humming. I mean, there's nothing in the acoustic literature that... you'd need at minimum a phonetically balanced...



Corn

Herman. Let it go.



Herman

The vocoder-per-relative clause alone...



Corn

Let it go.



Herman

Fine. Where was I. StyleTTS 2's TODO list, actually. Some items are still open, multi-speaker training code and a distributed training fix. Even the actively documented classics have loose ends. So that's the classic stack, and notice the pattern across all of it. Tacotron pinned to a dead framework, Coqui a fork of a dead company, Piper advertising for maintainers, Tortoise slow and untrainable by design, StyleTTS 2 excellent but



Herman (continued)

demanding four A100s for a hobbyist result.



Corn

So the newer generation is a response to exactly that graveyard.



Herman

It is, and the shift underneath it is bigger than any single model. The newer entries are large pretrained backbones with fine-tune scripts shipped alongside, which changes the practical question. It's no longer "can I train this." The pretraining already happened, on a hundred thousand hours of somebody else's audio. The question becomes "what does the fine-tune script assume about my data and my GPU."



Corn

And what the licence lets me do with the result. Start with the low-data darling.



Herman

GPT-SoVITS. MIT licence, and the claims are the boldest in the field. Few-shot fine-tuning from as little as one minute of data, zero-shot cloning from five seconds. And what makes it usable rather than just impressive is that the whole workflow lives in a WebUI. You slice your audio, optionally denoise it, let it transcribe automatically, proofread the transcripts, and then step through the fine-tune tabs. It holds your hand through the exact pipeline that, in the classic era, was a week of reading issues threads.



Corn

One minute of data. My nap snoring would qualify.



Herman

It's on a versioning spree, too. V2, V2Pro, V3, V4, which moved to forty-eight kilohertz output and fixed V3's metallic artefacts, and now V5. Inference is quick, real-time factor of zero point zero two eight on a 4060 Ti. The gaps are that training VRAM isn't stated



Herman (continued)

anywhere, and on Macs the GPU training path gives lower quality, so it falls back to CPU, which on a fine-tuning workload is a polite way of saying go make lunch.



Corn

The proofreading step in that WebUI. Is that load-bearing, or is it courtesy?



Herman

It's load-bearing, and hold that thought, because it's arguably the single most underestimated part of this entire field. We'll come back to it. Next, F5-TTS, which is the licence-mismatch case study of the newer generation. The code is MIT, the pretrained weights are CC-BY-NC, non-commercial, because the training data, the Emilia dataset, carries that restriction. So the same trap as XTTS-v2, one generation later, which tells you the trap is structural. Data licences flow downhill into weight licences.



Corn

So anyone fine-tuning F5 for anything commercial is inheriting the mismatch.



Herman

They are, and the fine-tuning itself is friendly enough. A Gradio app or accelerate scripts, your data a CSV of audio path and text, the V1 base model having been released in March twenty twenty-five. The docs contain one useful tip, which is that EMA weights, the smoothed moving-average weights, may hurt your early fine-tuned checkpoints, because early on they still mostly reflect the pretrained model rather than your voice. So try turning EMA off. That's the kind of advice you only get from someone who ran it and stared at the output.



Corn

The checkpoint that sounds like the pretrained model with a cold.



Herman

No data-size or GPU-memory guidance in the README at all, though, so you're on your own for sizing. Orpheus, next, is the Llama-backbone entry. It's literally a Llama three-billion model, Apache-2.0, pretrained on over a hundred thousand hours of English. The data guidance is concrete, quality shows after about fifty examples, and about three hundred examples per speaker is recommended for best results. Your data goes in as a Hugging Face dataset, training runs through their finetune script with accelerate, LoRA is supported, and the authors explicitly advise against synthetic training data, which is an interesting stance in twenty twenty-six.



Corn

Bold of them. Everyone else is drowning in synthetic everything.



Herman

The caution flags are that the smaller models, the one billion, four hundred million and hundred fifty million variants, are still listed as planned, the news items date to mid twenty twenty-five, and there are about a hundred and eighteen open issues. Healthy project or slow fade, I honestly can't tell you which.



Corn

OmniVoice.



Herman

The multilingual one. Apache-2.0, six hundred plus languages, zero-shot cloning from a three to ten second clip, plus voice design from attributes, where you describe the voice you want rather than supply one. Fine-tuning scripts ship in the examples folder, a main script and a LoRA variant, with your data as a JSONL manifest of id, audio path, and text. And the configs tell the whole story of this generation. The fine-tune starts from the released checkpoint for five thousand steps at a learning rate of five times ten to the minus five. The from-scratch run on the Emilia dataset is three hundred thousand steps at one times ten to the minus four. Sixty times fewer steps. That's what pretraining buys you.



Corn

Though the example config assumes two GPUs and says nothing about memory. So it buys you that, at an unspecified price.



Herman

Qwen3-TTS is the big-lab entry, released this January, Apache-2.0, sizes of zero point six and one point seven billion, in Base, CustomVoice, and VoiceDesign variants. And here's the limitation that matters, the fine-tuning folder in the repo supports single-speaker fine-tuning only, from the Base models, with a JSONL dataset and their own tokenizer. Multi-speaker is promised later. For a project whose entire brand is scale, the fine-tune path is the smallest door in the building.



Corn

And the awkward question, does the fine-tune actually beat just cloning?



Herman

That's the uncomfortable one. A test Baseten published reportedly found that fine-tuning on one and a half hours did not clearly beat zero-shot cloning on their measures. I want to hedge that, it's secondary, I haven't verified it myself. But it rhymes with what the StyleTTS 2 author admitted about one hour versus twenty-four. Fine-tuning is not automatically the upgrade people assume. If you've got ninety minutes of clean audio and a good zero-shot model, the fine-tune might buy you pronunciation quirks and consistency, or it might buy you nothing you can hear.



Corn

Which leaves the route with no price of admission at all.



Herman

Unsloth. This is the lowest barrier in the entire landscape, and I mean that literally, it runs in a free Colab notebook. They publish fine-tuning notebooks for Sesame CSM one billion, Orpheus three billion, Spark-TTS half a billion, Llasa one billion, Oute-TTS one billion, and anything transformers-compatible may work. The worked example uses a dataset called Elise, about three hours, roughly twelve hundred samples, single speaker, and on a Colab T4 a few epochs reportedly take one to two hours.



Corn

Three hours of audio, an afternoon on a free notebook. That's Daniel's CPU-only machine almost qualifying.



Herman

Almost. It uses LoRA at sixteen-bit precision and claims about fifty percent less memory, which is how it fits a three-billion-parameter model into a free tier at all. And this is worth dwelling on for one beat, because it reframes the whole classic-versus-newer story. Everything painful in the classic era, the A6000, the four A100s, the two and a half day runs, that was the price of learning phoneme structure from scratch. LoRA on a pretrained backbone says, don't relearn anything, just steer.



Herman

Now the correction segment, because the assumption most people carry is that the popular models are the fine-tunable ones, and that's backwards. Kokoro is eighty-two million parameters, Apache-2.0, hugely beloved for its size, and it is an inference library. No training section, no fine-tuning section. Voices are small tensor files. It's not a fine-tuning route as released.



Corn

The most famous small TTS model in the world, and you can't teach it anything.



Herman

Chatterbox is MIT with no official training code, and Resemble AI retired it in August twenty twenty-six, no longer maintained, left up as an archive. We covered that retirement in an earlier episode. And Fish Speech, or OpenAudio as it's now branded, is one of the most starred repos in the space, around thirty-four thousand stars, current models including the four-billion S2-Pro, and the licence is a custom Fish Audio Community License with the S2-Pro weights under a research licence. No fine-tuning guide I could find anywhere in the docs. Its cloning from ten to thirty seconds of reference audio is inference-time, not fine-tuning.



Corn

Three of the biggest names in the field, and the straight answer to Daniel's question is "not these."



Herman

Sesame CSM-1B rounds it out. Apache-2.0, Llama backbone with a small decoder emitting Mimi codec tokens, released in March twenty twenty-five. But the open model is a base generator, not tuned to any voice, and the README documents no fine-tuning. That happens through third parties like Unsloth. So the fine-tuning story of the newest model on the list is literally outsourced.



Hilbert

Corn. Those two thousand epochs at five hours. Was that clean audio, or audio with the silences already trimmed?



Corn

The guide doesn't say. Silence-trimmed, I'd assume, but that's an assumption.



Hilbert

Then I'll tell you what my brother-in-law says, and he does localization audio for a living, so weigh that however you like, he's not to be trusted on anything involving a bank. He says the framework is nothing. He says you could fine-tune any of them, the whole lot, on the same recordings, and get a usable voice. What kills it is the transcript. Every time, the transcript. Nobody proofreads. He goes on about this at length, usually at a table I'm trying to eat at.



Herman

That lines up with what Piper's own guide implies, that the validation score won't catch a wrong phoneme. You're saying the wrong phoneme comes from the label, not the model.



Hilbert

I found that out myself. I ran GPT-SoVITS, the V4 one, on about six hours of a man reading council minutes. Don't ask which council. The automatic transcription had written out the punctuation, rendered it as spoken words in the text file, and I checked the transcripts the way Herman checks a licence, which is to say once, at the end. The fine-tuned voice would say the word comma. Mid-sentence. Sad little voice, saying comma. I corrected every line by hand, retrained, and it came out clean. The model is gone now, the checkpoints were on a drive I gave away. I still have the corrected transcript. Forty-one pages. That's the part worth keeping.



Corn

You kept the transcript and gave away the model.



Hilbert

Obviously.



Herman

So the artefact with the human judgment in it survived, and the tensors didn't. The practical version is simple. A wrong label teaches the model the wrong sound on purpose, and that's data prep, which is its own conversation.



Hilbert

My brother-in-law also says never train on synthetic audio, but he also thinks Rome has a metro problem you can solve with bicycles, so.



Corn

He may be right on both counts.



Hilbert

Mm. The transcript thing he's right about.



Herman

So where does that leave Daniel's map. Classic routes, dormant or orphaned but conceptually clean. Newer routes, alive and cheap, but licence-mismatched or single-speaker-only or outsourcing their own fine-tuning. And underneath all of it, the actual skill has shifted. Nobody doing this well today is designing architectures. They're reading fine-tune scripts and weight licences.



Corn

Which is a strange thing to be good at, but it's now the whole job.



Herman

And notice what the Unsloth route quietly proves. The hard skill moved up a layer, to reading what the script assumes and what the licence allows.



Corn

If you take one thing from this episode, take the two licences. The code licence and the weights licence are two different documents, and the one that governs the voice you make is the one nobody reads. XTTS-v2, F5-TTS, both permissive code sitting on restricted weights.



Corn

The open question I'd leave with Daniel is sustainability. Every classic route is either dormant or advertising for maintainers, and every newer route has a catch stitched into it. Ask me in a year which of these stacks you'd actually build on, and I don't know whether the answer exists yet.



Herman

My guess is the survivor looks like Unsloth's approach, a thin fine-tuning layer over other people's pretrained backbones, with the licence diligence as the actual craft.