Episode Book

EPISODE #6682

Context-Driven LoRA Mixing for Speech Recognition

PUBLISHED	19 November 2026 at 13:11 UTC
DURATION	23:50
LISTEN	myweirdprompts.com/episode/speech-lora-adapter-blending-32705/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
BOOK GENERATED	19 November 2026 at 13:34 UTC

SYNOPSIS

No published speech recognition system lets an AI agent dial LoRA adapters—like eighty percent tech jargon and twenty percent code-switching—from app context, and those percentages aren't real weights anyway. The discussion unpacks why NeMo's adapter API only switches or chains bottleneck adapters, why encoder LoRA hurt Parakeet-TDT in a dysarthric-speech study, and how confidence arbitration in DAS, decode-time biasing, and LLM tools like PEFT, TIES, and X-LoRA actually handle multi-adapter composition.

TRANSCRIPT



Corn

He kept one model. That's the detail I keep turning over, Herman. Daniel trained a code-switching fine-tune, then a tech-jargon one, expected each to win its own ballpark and the combination to lose both, and the combination lost nothing. So Sabich stayed a single model, because he'd rather live with one than switch between two or three.



Herman

And now the LoRA episodes have been gnawing at him.



Corn

His what-if is the road not taken: one base ASR model, a LoRA adapter for tech vocabulary, another for the Hebrew–English switching, a UI that presents them as separate models, "I'm using the English version" — and the part he finds amazing, an AI agent setting the adapter balance from context. Working in a code repo, tech vocabulary at eighty percent strength, code-switching at twenty. A Hebrew voice note comes in, the dials flip. He wants to know whether any methodology like this exists, and whether it's feasible.



Herman

Then let's start with what people have actually built, because the pieces are more scattered than you'd think. And some of them don't fit together the way the phrase "adapter balance" implies.



Corn

Go on.



Herman

Daniel isn't really asking whether there's a paper with this exact title. He's asking whether adapter composition is mature enough in speech that context-driven mixing is the obvious next step, or whether ASR is behind the LLM world here. And the short answer is: nobody has published an ASR system where an outside context signal sets continuous adapter strengths at inference. I've looked, and I don't think it's out there. But every individual piece exists, and the LLM world has already shipped the "different models in



Herman (continued)

the UI, one model underneath" pattern.



Corn

So the answer is no, surrounded by yeses.



Herman

That's the shape of it. The useful thing to hold onto is that Daniel's eighty-twenty is conflating two different knobs. One is a weight on an adapter — how much of its learned change gets added to the base model. The other is a decode-time bias — how hard a phrase list pushes the model's hypotheses toward certain words. Daniel's slider sounds like the first knob. The thing that actually works today is much closer to the second. Keep that split in your head and the whole landscape sorts itself out.



Corn

Two faders, one labeled weight and one labeled bias, and his idea assumes they're wired to the same bus.



Herman

They're not even on the same desk. Start with the standard methodology, because it's less glamorous than he hopes. NeMo — which is what Sabich is built on, Parakeet with a FastConformer encoder and a TDT transducer — has an adapter API. You can add an adapter, enable a set of them, save them, load them. But those are residual bottleneck linear adapters, not LoRA. And when you enable several, NeMo chains them — the output of one feeds the next. There is no per-adapter strength parameter anywhere in the documented API.



Corn

So the eighty-twenty slider arrives and finds no socket in the wall.



Herman

Nothing to plug into. What NeMo does support is per-request on and off switching — enable this adapter, disable that one, per request. Disabling all of them gets you back to the base model exactly, as long as the batch-norm statistics were frozen during training, which they normally are for adapter training. So the supported model of "several adapters" is a chain, and the supported model of "using them" is selection. Blending is simply not in the API.



Corn

And I'd guess there's no official LoRA recipe for Parakeet, since you're dwelling on bottleneck adapters.



Herman

Correct, and I checked this more than once because it surprised me. NeMo's LoRA and DoRA recipes target language models. For Parakeet and FastConformer, nothing official. Which makes the community results the only signal, and the strongest one is a warning sign. September this year, a single-speaker dysarthric-speech study — one Hungarian speaker, four hundred and nine utterances — put LoRA rank sixteen on Parakeet-TDT-0.6B-v3's encoder.



Corn

And it went how badly?



Herman

Character error rate went from 50.6 percent to 69.1 percent, and the model started hallucinating English and Russian fragments. In the same study, Whisper large-v3 and Qwen3-ASR both improved with LoRA. The authors' hypothesis — untested, they say so themselves — is that the encoder LoRA disturbs the language prior baked into the transducer's joint network.



Corn

One speaker, so hold it lightly.



Herman

I hold it at arm's length, yes. But it's the only direct data point we have on LoRA onto a transducer-class model, and Sabich is a transducer. If that result generalizes even slightly, Daniel's plan of a LoRA per domain on Parakeet is starting from a weaker foundation than the LoRA discourse would suggest. Whisper behaves like a good citizen; the transducer may not.



Corn

So the first honest thing to tell him is that the "one LoRA per domain" half of his idea is the half with no recipe and one bad data point, not the routing half everyone would instinctively doubt.



Herman

That inversion is the episode, basically. Because the per-domain adapters themselves — the boring, unglamorous half — are real and working in research. Just never with continuous weights. LoRA-Whisper, from 2024, trains a language-specific LoRA per language on the encoder and decoder, base frozen, which cuts cross-language interference and makes adding a language cheap. Then there's the Interspeech 2025 work on LoRA language experts — separate experts per language, fused or distilled into one student — and it reports about a ten percent relative gain over standard fine-tuning when the system knows the language, about fifteen percent when it doesn't.



Corn

A gain for not knowing more.



Herman

It's a nice result, because the language-agnostic setting is the realistic one — audio comes in, nobody labels it. And that points at the piece I find most elegant: a hierarchical LoRA mixture of experts on mHuBERT, from January this year. A shared LoRA plus language-specific experts, and the routing is done by the model's own language-ID posteriors taken from the audio. No label needed. The model's own internals say which expert to weight.



Corn

No, I see why you like that one. The router is inside the model, reading the audio, and it already exists.



Herman

The Mandarin–English code-switching mixture-of-experts work from 2024 does the same thing, and Zipper-LoRA for speech LLMs, from March. Route on language identity inferred from the audio. That's the pattern across all of it — and here's the thing I want to name plainly. In every one of these, the router reads the audio. Daniel's idea inverts the signal. The app, the repo, the message thread tells the model what it's about to hear, before a single sample arrives.



Corn

Which is information the acoustic router can never have, because it has to wait for the audio.



Herman

That inversion is the novel part of his idea, and it's also the part nobody has published. Nobody is pulling the routing signal from outside the model.



Corn

You realize the sympathetic reading is that Daniel wants the microphone to know which window is open.



Herman

The window being open is cheap, reliable context. The audio has to guess. So before we crown him, the closest thing to several adapters live at once in ASR — and it's worth a proper look, because it's the only design I found where that's true. It's called DAS, January 2025. Three LoRA domain adapters on Whisper-base — Music, Weather, Sports — trained on TTS audio of LLM-written text. At each decoding step, every adapter proposes a token, and the most confident one wins, provided it beats the base model by a margin.



Corn

Every step, a small election.



Herman

A per-token election, yes. Nobody sets weights. No domain signal is used. It's pure confidence arbitration. And it works — word error rate goes from 27.9 to 24.9 on Music, 15.0 to 12.4 on Weather, 15.6 to 14.0 on Sports, roughly ten to seventeen percent relative, while losing only about one percent out of domain on LibriSpeech.



Corn

Those are real gains for a system with no idea what it's listening to.



Herman

And the cost curve is instructive. Real-time factor of 0.024 for the base, 0.027 with three adapters, 0.029 with ten, 0.045 with twenty-five. Cheap at three, still under twice the base at twenty-five, but it grows with every adapter. Daniel wants two, so cost is a non-issue.



Corn

So DAS answers the mechanism question honestly: several adapters can be simultaneously live, but what arbitrates them is confidence, not strength. There is no percentage anywhere.



Herman

Which is the answer to the literal question, "is there a methodology like this." The methodology that exists is selection, chaining, and confidence arbitration. Continuous weighting by outside context — no. And before we get into whether it could exist, the eighty-twenty question deserves its own autopsy, because I don't think it means what it sounds like it means.



Corn

I was hoping you'd say that, because my instinct all week has been that "eighty percent" is a number with a decimal point's confidence and none of its meaning.



Herman

Here's the problem. LoRA weights from different fine-tunes are less aligned with each other than full fine-tunes are. Naively adding them makes them interfere. That's the entire reason TIES and DARE exist — those are merging methods that trim and resolve conflicting parameters before summing. And there's a paper called KnOTS, October 2024, that aligns the LoRA matrices by singular value decomposition before merging and gains up to 4.3 percent over naive addition. Four percent, from just putting the pieces in the same coordinate system first.



Corn

Which tells you the pieces were not in the same coordinate system.



Herman

It tells you the default is that they aren't. Now, Hugging Face's PEFT library does have add_weighted_adapter — you name the adapters, you set weights, it builds a merged adapter, with linear combination or TIES or DARE-TIES underneath. So the tool exists, for text models. But look at what the docs tell you: start with all weights at 1.0, and note that weights above 1.0 often work better, because they preserve the scale of the update.



Corn

Above one. The knob's own manual admits the numbers aren't percentages.



Herman

They're relative magnitudes in a sum, not a share of a whole. There's no semantics where 0.8 plus 0.2 equals one meaningful model. And here's the kicker for Daniel: no ASR study I know of maps word error rate as a smooth curve over mixing weights. Nobody has drawn the dial's response. So "eighty percent tech, twenty percent code-switching" is a calibration problem, not a setting.



Corn

Meaning the number is aspirational until you've held it against your own eval set and seen what it does.



Herman

That's the honest shape of the knob. The closest ASR result to a scalar dial is task arithmetic on Whisper, from January this year — you take a task vector from a high-resource support language and add it to a low-resource language with a single scalar weight, tuned on validation error. Up to ten percent improvement across twenty-three low-resource languages. Note the verb, though: tuned. Someone searched for the coefficient on a validation set. Nobody read it off as a percentage.



Corn

So even the best version of the dial is a dial you find by experiment, not one you set by intent.



Herman

Now, the LLM side has gone further, and it's worth seeing how far, because it tells Daniel what the ceiling looks like. LoraRetriever, February 2024, retrieves and composes LoRAs according to the input prompt — that's the closest published thing to outside context picking the adapters. X-LoRA learns a gate on the hidden states that sets each LoRA's strength per token and per layer.



Corn

That's his slider, on text.



Herman

Almost — but the signal driving X-LoRA's gate is internal. The hidden states, not the repo path. LoraHub, a bit earlier, finds mixing weights for an unseen task from a handful of examples by gradient-free search — per task, offline, not live at inference. Nobody is sliding the fader mid-conversation based on which app has focus.



Corn

The serving side, though — that's where I suspect the story is actually done.



Herman

Completely done. S-LoRA, from LMSYS, November 2023, serves up to two thousand adapters on a single A100 with eighty gigs, batching requests that each use a different adapter, up to four times vLLM's throughput. But read the fine print: one adapter per request. The system picks a lane; it never straddles two.



Corn

Which is precisely the "different models in the UI, one model underneath" pattern, in production, for LLMs — just without the mixing.



Herman

Without any mixing at all, and that's not an accident, I think. So here's how I'd translate all of it into Daniel's architecture. A small router in front of the ASR service that reads cheap context — active application, repository path, file extension, the language of the message thread, maybe a user toggle — and maps it to a preset. Code repo: tech adapter on, code-switching off. Hebrew voice note: the reverse. Mixed: both on, in a chain, or DAS-style arbitration. The router is a classifier, or honestly, a rules table. It's an afternoon of work.



Corn

So the agent in his vision is the easy part. He could build it this weekend and the hard part wouldn't even notice.



Herman

The hard part is the adapter side, which we've just spent half the episode discovering has no strength knob and possibly a hostile transducer under it. But here's the punchline, and I think this reframes his whole idea: the context knob he wants already exists in ASR. It's just not attached to adapter weights. NeMo's GPU phrase boosting applies at decode time, no retraining, works on CTC, transducers, TDT, Canary, greedy or beam. It has a tunable strength — the boosting alpha and a context score. And it supports per-utterance phrase lists: each request in a batch can carry its own biasing list, including greedy TDT and cache-aware streaming.



Corn

Per request. So an agent could send the "code repo" list or the "Hebrew voice note" list today, with zero weight changes.



Herman

Today, this afternoon, on the stack he already runs. TurboBias, which NVIDIA put out at ASRU last year, handles up to twenty thousand key phrases without noticeable speed loss, and it's open-sourced in NeMo. On Whisper's side, prompt biasing work from 2025 cut rare-word error from 23.7 to 18.0 percent and out-of-vocabulary error from 60 to 37, zero-shot. Though a benchmark from December 2025, ProfASR-Bench, adds the sober note: prompted systems often barely use the context even when handed the gold transcript.



Corn

Handed the answer key and still daydreaming.



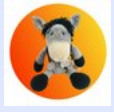
Herman

Some of them, yes. And the limit that matters for Daniel: biasing only reweights hypotheses the model can already produce. It can boost "OPNsense." It cannot teach the model code-switching acoustics, or an accent, or how Hebrew actually sounds mid-English-sentence. That's what the fine-tune is for. Which is exactly why his combined Sabich was the right call for acoustics, and why biasing lists on top of it are the right call for vocabulary.



Corn

Before we get to whether he should go back to separate adapters at all, there's a household question. Doesn't he also run this thing on a phone?



Herman

He does, and the on-device story is a reality check. sherpa-onnx ships Parakeet-TDT v2 and v3 as int8 ONNX with Android builds, but there's no documented way to load LoRA or adapters at runtime. ONNX toolchains generally merge adapters before export. So on a phone, in practice, each preset is a separately exported merged model — unless you go engineer adapter inputs into the graph yourself, which is a real project. And NeMo's transducer export fuses the decoder and joint while sherpa-onnx expects them separate,



Herman (continued)

so his fine-tuned Parakeet already needs a dedicated export step even before adapters enter the picture.



Corn

His elegant two-adapter phone setup is, on the ground, one exported model file per preset, pretending to be one philosophical position.



Herman

For now, yes.



Hilbert

Medieval Pottage Facts. Many peasant rents were paid in hens and eggs, handed over to the lord rather than eaten by the family. The lord ate. The family watched.



Corn

Nothing like rent to put a fine point on who eats.



Herman

So when do separate adapters actually beat one merged model? Because Daniel's own result — combined lost nothing — is a strong default. The cases where I'd go the other way: many domains accumulating over time, where retraining one combined model per addition gets expensive. Other users who want only the public capability — say, someone wants the code-switching vocabulary without any of the rest. A personal-voice adapter kept private and separate from public vocabulary adapters, because a merged checkpoint cannot be split back apart. And server-side per-request swapping, the S-LoRA pattern. On a server, two adapters and a router is a nicer shape than two full models.



Corn

That last one lands with me, the un-splittable merged checkpoint. Daniel's voice data staying its own object, separable, has a value that no error rate measures.



Herman

And it's the one argument for adapters that survives even if the mixing dream dies.



Corn

Which leaves the verdict, and I want to state it plainly so Daniel gets a clean answer. Methodology: no, not as he described it — no published system where outside context sets continuous adapter strengths. Feasible: yes, in a modified form. A small router picking one of two or three adapters or presets per request is entirely realistic, and per-request biasing lists give him "context sets the balance" for vocabulary at almost no cost. The eighty-twenty dream is achievable as a calibrated preset table, not as a live percentage.



Herman

Calibrated being the operative word. He'd tune each preset on his own eval set, the same way the task-arithmetic people tuned their scalar. The numbers on the presets would be meaningless to anyone else, which is fine, because the numbers were always meaningless.



Corn

Which means the answer to his question turned out to be: mostly the shape he already chose, one model, plus a cheap context layer on top.



Hilbert

Yamaha 01V96. Eight hundred quid, 2009, from a bloke in Peckham. My brother-in-law Ray does live sound for small venues and he still runs one, refuses to touch plugins. Every channel has a physical fader and he keeps a laminated card taped to the desk with the fader positions written on for each act. House band, vocals at seven, kick at six, the horn player at four because he leans in. He sets it all by hand before the first song and doesn't touch it again unless something goes wrong. Tried a digital desk once. Hated it. You can't feel where the fader is without looking. Corn, you'd get on with him. He'd say an eighty-twenty slider is a plugin with a hundred knobs. I re-did his card once, put it in setlist order instead of desk order. He bin'd it. The card's order is the desk's order, not the



Hilbert (continued)

set's. I still think mine was tidier.



Herman

Ray's card is doing adapter routing with a rules table. Your version was a context-indexed lookup, and he wanted the hardware-indexed one.



Hilbert

Herman, the desk is the hardware. The setlist changes every night. The desk doesn't. He does a full changeover in eleven minutes and the card is why.



Corn

The card also explains something the papers don't. Under pressure, a percentage is unusable — Ray went straight to a small number of positions you can hit blind.



Hilbert

He'd have made a decent engineer. Terrible at email.



Corn

Fair. So where does that leave the open questions? Two, for me. If the router is the easy part and the adapter side is the hard part, what would it take to make adapter strength a calibrated, measurable quantity in ASR, the way it sort of is in LLM merging? Someone has to draw the error curve over mixing weights, and nobody has.



Herman

And the second one nags at me more, honestly: does the context signal Daniel wants to route on — app, repo, thread — actually predict the acoustics well enough to be worth routing on? The mixture-of-experts people route on the audio's own language posteriors because that signal is already about the sound. A repo path is a proxy. The proxy might be worse than the model's ears.



Corn

Which suggests the final answer may be both signals — context to pick the preset, audio to catch the proxy being wrong. The pieces are all shipping separately. The glue is the part nobody has published.



Herman

Produce that glue and you're the whole paper.



Corn

Thanks as always to our producer, Hilbert Flumingtop, at the desk through all of it. For more along these lines, there's episode five, Fine-Tuning ASR For Maximal Usability; episode seven, Building Custom ASR Tools; and episode nine, Benchmarking Custom ASR Tools - Beyond The WER. This has been My Weird Prompts. If today's question earned its keep, leave us a review wherever you listen — it's how the obscure ones find us.



Herman

See you tomorrow, everyone.