Episode Book

EPISODE #5868

Fine-Tuning Parakeet for Um-Free, Hebrish Speech

PUBLISHED	11 October 2026 at 13:37 UTC
DURATION	27:36
LISTEN	myweirdprompts.com/episode/fine-tuning-parakeet-hebrish-speech-40484/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.3
SOURCES	18 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

Daniel wants to bend Parakeet to his will: strip the ums, catch niche words, and handle the Hebrew he sprinkles into his English. He's done this before with Whisper, so he knows the shape of the job — but Parakeet is a different animal. This episode walks through what Parakeet TDT 0.6B v3 actually is, why a small model is a better inference candidate but not an easier fine-tune, and the real workflow: NVIDIA NeMo on a rented GPU, a JSON manifest, two learning-rate knobs, then a GGUF conversion and a sideload back to the phone. We dig into the disfluency problem (there's no published Parakeet recipe, but a Whisper LoRA shows the trick: the lesson lives in the target transcript, not in a rule), the Hebrew code-switching problem (why the obvious Hebrew Parakeet model is the wrong tool, and what a Hebrish Whisper fine-tune got right with just 500 phrases), and the tokenizer-extension mechanics that matter if you ever need to teach the model new words without forgetting the old ones.

TRANSCRIPT



Corn

How long does a fine-tune stay good for?



Herman

That's the question underneath Daniel's whole message, and I don't think he knows he's asking it.



Corn

He's been running Parakeet on his OnePlus for a month. First on-device speech model he says matches what he gets from the cloud. And now he wants to bend it. Drop the ums, catch the niche words, handle the Hebrew he sprinkles into hise English.



Herman

And he's done this before with Whisper, so he knows the shape of the job.



Corn

Right, the shape. He generated sentences with an AI tool, recorded them one by one, paired the audio with ground truth, fine-tuned on that. So his question is: what's the equivalent for Parakeet. And then two follow-ups. How much data before he's just overfitting. And when Parakeet v4 ships, does he start over, or keep training from his first checkpoint.



Herman

That last one is the interesting one.



Corn

It's the one he's most confident about, too. He's already planning the incremental path.



Herman

That's where I'd push back. But before we get to the pushback, there's a premise in his message that's worth untangling, because it's the kind of thing that sounds obviously true and isn't quite.



Corn

The small model.



Herman

He says Parakeet is small, that's why it doesn't need heavy quantization the way Whisper does on a phone, and so it should be a particularly good candidate for fine-tuning. Two of those three clauses are right.



Corn

And the third one sends him to a GPU.



Herman

It does. Let me lay out what Parakeet actually is, because the architecture is the reason it runs on his phone at all. Parakeet TDT 0.6B v3 is six hundred million parameters. Multilingual, twenty-five European languages, and it auto-detects language without being told which one it's hearing. Version two was English only. Same architecture family, bigger language coverage.



Corn

And the TDT part.



Herman

Token-and-Duration Transducer. A normal transducer predicts the next token. TDT predicts the token and how long it lasts, so it can skip the blank frames instead of stepping through them one at a time. NVIDIA's own numbers put it at sixty-four percent faster than the Parakeet RNNT 1.1B that came before it.



Corn

Sixty-four percent faster, and it's smaller. That's why it fits in his pocket.



Herman

That's the whole reason. The decoder family matters more than the parameter count, honestly. A TDT or CTC decoder runs at ten to a hundred times the throughput of the big LLM-decoder models, and gives up fractions of a point of word error rate to do it. So you get a model that fits unquantized-ish on a phone and still transcribes well.



Corn

Unquantized-ish. That's a word he'd want on a t-shirt.



Herman

It's the honest word. He's running GGUF weights, so there's some quantization in there somewhere, but nothing like the compression you'd need to squeeze a Whisper large into the same phone. Parakeet at q8 is about thirty-seven percent of the f32 size, and it's still smaller than Whisper's int8.



Corn

So the small-model instinct is half right.



Herman

Half right. A small model is a better inference candidate. It is not an easier fine-tune. Fine-tuning Parakeet happens in NVIDIA NeMo, on a GPU, with the full checkpoint, and the phone is not in that loop at all. The phone runs inference and nothing else.



Corn

So the honest shape of this is: train on a rented GPU, export a GGUF, sideload it.



Herman

That's the workflow. It's not onerous, but it's not what he pictured. And there's a subtle benefit to it. Because the base model is small, the fine-tuned version is still small. A six-hundred-million parameter model with a modest adaptation still ships as a file he can carry around. If he were doing this with a Whisper large, the fine-tune would be a two-gigabyte problem every time.



Corn

Alright. Take me through the actual recipe.



Herman

NeMo has a fine-tune script, speech to text finetune, with a YAML config next to it. You point init from pretrained model at the Hugging Face id for Parakeet v2, or init from nemo model if you've got a local dot nemo file sitting on disk. The training data is a JSON manifest. Each line has the audio file path, the transcript, and the duration. That's it. That's the whole input format.



Corn

And the knobs.



Herman

Two that matter. Epochs, typically fifty to a hundred for domain adaptation. And learning rate, somewhere between one times ten to the minus four and one times ten to the minus five. The docs are blunt about it. Start low. Fine-tuning with too high a learning rate destroys the pretrained features, and then you've spent a GPU hour to make the model worse.



Corn

There's a scheduler gotcha in there too.



Herman

There is. The default scheduler is Noam, which has a warmup phase. On a fine-tune, that warmup resets your learning rate to something high right at the start, which is exactly what the docs told you not to do. So you override it. Constant, or cosine annealing. The docs say so explicitly.



Corn

Constant or cosine. So the model doesn't get thrown in the deep end on step one.



Herman

And one more thing for anyone changing the tokenizer, which is going to matter for the Hebrew half of his project. The docs say exclude the decoder and joint from initialization. That keeps the pretrained encoder and lets the decoder learn the new vocabulary without dragging the acoustics along with it. That line is going to come back.



Corn

Then the trip home. NeMo checkpoint to phone.



Herman

There's a converter in parakeet.cpp. Convert parakeet to GGUF takes a NeMo checkpoint, either straight from Hugging Face or a local dot nemo, and writes GGUF. There's a second script for Hugging Face safetensors if that's what you've got. Then quantization is a separate step. You run the parakeet CLI with quantize and pick your level. f32, f16, q8, q4. So the round trip is: NeMo in, dot nemo out, convert to GGUF, quantize, copy to the phone.



Corn

He's done that part at least once, since he's already got v2 running.



Herman

The conversion is the easy half. Now the actual target. He wants the ums gone.



Corn

This is the one where I expect a paper and instead you're going to tell me it's a Whisper LoRA.



Herman

It is a Whisper LoRA. There's no published Parakeet disfluency recipe. What there is is a Whisper small model with a smoothed LoRA on it. Rank thirty-two, alpha sixty-four, dropout oh five, targeting the query and value projections, three and a half million trainable parameters. About one point four percent of the model.



Corn

And it did what.



Herman

Word error rate on disfluent speech went from twenty-six point oh four percent to twenty point four six. Twenty-one percent relative improvement, trained on about twenty-seven hours of data, ninety-four hundred utterance pairs, three epochs, on an M1 Max in about two and a half hours.



Corn

An M1 Max. He doesn't need a data center.



Herman

That's the encouraging part. The interesting part is how they built the targets. The smoothed transcripts were made by dropping the filled pause markers. The ampersand uh, the ampersand um. And then collapsing duplicate words. So the target text is the ground truth that teaches the model to leave them out.



Corn

Say that again, because that's the trick.



Herman

The lesson isn't a rule you give the model. It's in the target. If the audio says um and the transcript doesn't, the model learns that the um is furniture. It's in the room and it isn't part of the sentence.



Corn

He already knows how to build pairs. He did it for Whisper.



Herman

And that method transfers exactly. Record, keep the audio, write the transcript the way he wishes it had come out. No ums. That's a dataset he can build in an afternoon.



Corn

The Hebrew is where it gets interesting.



Herman

It's where the design fork is, and I want to get this right because it's the part Daniel would get wrong if he picked the obvious route. There's a Hebrew Parakeet model. Fine-tune of v3, and the recipe is the most detailed public Parakeet adaptation anyone's posted. And if he just grabs it because it's Hebrew, he's made a mistake.



Corn

Because it's Hebrew-only.



Herman

The model card is blunt. English words come out in Hebrew letters. It was built to transcribe Hebrew speech, and it does that well, and anything English you feed it gets chewed up and spat back out in the wrong alphabet.



Corn

He wants the opposite.



Herman

He runs English-primary. His dictation is English with Hebrew words dropped into it. Makolet. Teudat zehut. That shape is not what the Hebrew model was built for, and using it would be worse than his current situation.



Corn

So the model he actually wants to imitate is the one you named earlier. The one from a blog post.



Herman

The Whisper Hebrish fine-tune. Same problem, different model. Someone wanted Whisper to handle English with Hebrew sprinkled in, so they asked Claude to generate a CSV of five hundred words and phrases common among English-speaking Israelis. Recorded sentence-level pairs. Fine-tuned Whisper Large V3 Turbo on a Modal A100.



Corn

Five hundred phrases.



Herman

Five hundred. And the result is the demo I'd want to hear. Stock Whisper transcribes the sentence as Macaulay, and Theodette Sahoot. The fine-tune gets makolet and teudat zehut.



Corn

So it works at that scale.



Herman

It works at that scale, for that pattern. Code-switched words are exactly the thing a monolingual model drops, because they don't sound like the monolingual data. That's the useful reframe. The model isn't failing to understand Hebrew. It's failing to expect it.



Corn

Alright, so the tokenizer. You said that line would come back.



Herman

Here's where it comes back. That Hebrew Parakeet recipe, the one that's Hebrew-only and wrong for Daniel, is still the best guide to the mechanics, and its central instruction is: extend the tokenizer, don't replace it. Train a Hebrew BPE on about two hundred fifty hours of audio, then append the new pieces after v3's existing eight thousand one hundred ninety-two tokens. The vocabulary goes from eight thousand one hundred ninety-two to nine thousand two hundred six.



Corn

And the old weights.



Herman

You keep every pretrained weight by copying the old rows back by token id. Old ids and merges stay intact, and you bolt the new vocabulary onto the end without disturbing what's already learned.



Corn

So the model can spell new words without forgetting the old ones.



Herman

Not forgetting them in the tokenizer, at least. And there's a learning-rate detail that pairs with it. Ten times learning rate on the decoder and joint. Two times ten to the minus three, against two times ten to the minus four on the encoder. The new tokens start identical and untrained, so Adam moves them at about the learning rate per step. At a normal rate they'd stay indistinguishable for thousands of steps. That's the long empty-output phase people hit when they extend a tokenizer and then wonder why the model says nothing.



Corn

And the encoder stays slow.



Herman

The encoder stays slow. One times ten to the minus five to three times ten to the minus five. Short warmup, bf16, roughly six hundred seconds of audio per optimizer step. There's one more gotcha I want in the notes, because it's the kind of thing that eats a day. Set the greedy cuda graph decoder to false for in-training validation. If you don't, the validation pass decodes with stale weights and shows empty output, and you spend an afternoon convinced your fine-tune is broken.



Corn

It isn't broken, it's just reading yesterday's model.



Herman

It's reading yesterday's model. Turn the flag off, validation works, life continues.



Corn

So the shape of his answer. Disfluency: Whisper recipe, transfer the method. Code-switching: Hebrish-style pairs, English base, don't grab the Hebrew model. And the tokenizer-extension detail is a Parakeet-specific bonus he hopefully never needs.



Herman

He hopefully never needs it, because a good Hebrish fine-tune wouldn't need a tokenizer change at all. Makolet is latin letters. He's not adding Hebrew script to the vocabulary, he's adding romanized Hebrew words that already tokenize fine. He just needs the model to expect them.



Corn

Which is a much cheaper fine-tune.



Herman

Much cheaper. Now, data volume. He asked for a decent size, and the honest answer is that nobody has published a number for Parakeet on this exact task.



Corn

Say so plainly, then. He can take honesty.



Herman

So here are the three reference points that exist. Five hundred generated phrases for Hebrish. Twenty-seven hours, ninety-four hundred pairs for the disfluent LoRA. And about nine thousand hours of pseudo-labeled audio for the Hebrew Parakeet.



Corn

Nine thousand hours is a different species of project.



Herman

It's a different species. That's full language adaptation. It's the number you'd need if you were building the model Daniel should not use. His job is personalization. Hundreds of recorded sentences, not thousands of hours.



Corn

So Hebrish scale.



Herman

Hebrish scale is the realistic template for what he described. A few hundred sentences, recorded one by one, with the transcripts the way he wants them. He's already done this pipeline once. He's not starting from zero on the recording side.



Corn

Now the trap. He wants incremental training from his first checkpoint, and you've been sitting on the answer.



Herman

The literature is unanimous against the convenient path. June of this year, a paper called Learning to Hear Hesitation, which is doing exactly his task, disfluency-aware training. And it says adapting models on limited datasets can lead to catastrophic forgetting of general-domain knowledge. It even names a trade-off between marker learning and ASR performance. The model gets better at leaving out the ums, and worse at everything else.



Corn

That's the whole game. He pays for one improvement with ten regressions he didn't ask for.



Herman

And then there's the continual-learning paper that gives the actual numbers. Naive sequential fine-tuning against two mitigation methods. Elastic weight consolidation cut relative word error rate by five point two one percent versus the naive baseline. Synaptic intelligence did four point three six percent. So the mitigations work, and the naive path is the one they beat.



Corn

And there are newer ones.



Herman

Two from January. Inverse-Hessian regularization, and rehearsal via SVD. Both memory-free or cheap-memory, both aimed at exactly his scenario, where you don't want to keep the whole old dataset on disk just to not forget it.



Corn

So when Parakeet v4 ships, what does he do.



Herman

He re-runs from the new base model. Same dataset, fresh fine-tune. Or he rehearses. He keeps a slice of the old pairs in the new training run, maybe twenty percent, so the model refreshes the old behavior while it learns the new one. What he should not do is train straight from his v2 fine-tune checkpoint onto v3 with only the new data in the mix. That's the naive sequential path. That's the path those numbers are measuring.



Corn

And the honest word for the incremental path he's planning.



Herman

It's a research problem he's about to run on himself, without the baseline to know he lost anything. If the ums get better and the rare words get slightly worse, how would he ever notice? He doesn't have a test set. He's got his own ear, and his ear is going to notice the ums.



Corn

So he'd build a small eval set with the same care he builds the training set.



Herman

That's the answer. Hold out fifty pairs. Run them before and after. If the old ones are still right, train from the checkpoint. If they aren't, you've got an answer the literature already gave you, and you overwrite.



Corn

That leaves the boring option, which I want on the table because it fixes his favorite example with no training at all.



Herman

Word boosting. NeMo has it. GPU-PB, CTC-WS. You give it a list of words and phrases, and it biases the decoder toward them at inference time, no retraining. It is precisely the fix for Corn becoming Corinne. And for that short list of niche vocabulary. And a new word takes effect in seconds.



Corn

But.



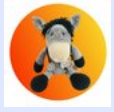
Herman

But it's a NeMo decoding-time feature. It lives in the GPU inference path. It is not in the GGUF runtime on his phone. parakeet.cpp doesn't have it, sherpa-onnx doesn't have it, and neither of those is going to grow it because he asked.



Corn

So the fix for something bearing my name is locked behind a GPU.



Herman

It's a beautiful design artifact. The easiest fix for his most annoying daily problem is one he can't reach from the device that's actually bothering him.



Corn

He said as much. The one I'm using doesn't have a personal dictionary. So he's fine-tuning because the tool he wants doesn't exist yet.



Herman

And that's the real shape of his problem, bigger than Parakeet. He's doing personalization on a model that wasn't designed to be personalized. The features that would make it easy, word boosting and on-device adaptation, either need a GPU or don't exist in the runtime he's shipping on. So he's substituting a research-grade fine-tune for a missing product feature.



Corn

So the whole episode is really about the gap between running a model and owning it.



Herman

When he runs Parakeet, he's renting a behavior someone else chose. Fine-tuning is the only way to own it, and the ownership comes with a maintenance burden he didn't sign up for. Every new base model is a fresh decision.



Corn

Herman, before you ask. What does the tokenizer-extension recipe mean for the fact that his disfluency and Hebrew data live in one dataset?



Herman

That they should probably be separate fine-tunes, or one fine-tune with a mixed dataset and a very careful learning rate. But I'll be honest, I'm not sure which. Mixing a token-level task, which is what disfluency removal is, with a vocabulary task, which is what Hebrish is, in one run, gives you a loss curve that's averaged across two very different things. There's no published recipe for that combination. I'd want to see it before I recommended it.



Corn

So he might be looking at two fine-tunes and a merge.



Herman

He might. And he'd want to measure both, which is the same advice as before. Hold out the pairs. Check before and after. Nobody's going to do that for him.



Corn

Let me ask you something.



Herman

Go on.



Corn

What is the smallest dataset that would actually work for the disfluency piece alone. Forget Hebrew, forget niche words. Just ums.



Herman

I'd guess a few hundred pairs would move it. The LoRA used twenty-seven hours, but that was a general disfluency model, meant to work for anyone. He's one speaker, one microphone, one accent. A few hundred of his own sentences might do as much to his own audio as ninety-four hundred pairs did to a crowd.



Corn

Because the model doesn't need to learn um removal in general. It needs to learn that his ums aren't in the transcript.



Herman

If he'll excuse the word for a second, yes. The task gets narrower when it's personal. That's the one place his scale-up instinct might be wrong in the helpful direction.



Corn

And how much data would make overfitting worse rather than better.



Herman

When it stops generalizing. When he holds out twenty of his own recordings and the fine-tune does worse on those twenty than the base model. The number that causes that is different for every voice. There isn't one.



Corn

Which is the same answer he got on the dataset size.



Herman

It's the same answer because it's the same question. The literature gives analogues, and his own held-out set gives the truth.



Corn

And he mentioned the fine-tune bloating the weights.



Herman

He did. He said the variant he uses is very fast, so even if the fine-tune bloated the weights a little, the inference is fast enough that it would be fine. That instinct is right, and the reason it's right is the tokenizer. If the fine-tune changes the vocabulary, say for a Hebrew-script tokenizer extension he hopefully won't need, the embeddings and the output layer grow with the new token count. Eight thousand one hundred ninety-two to nine thousand two hundred six is a twelve percent increase on those two matrices. Everything else stays put. So a Parakeet fine-tune in the Hebrish style, which adds no tokens at all, is close to the same size as the base model.



Corn

So the fine-tune that fits his goal is also the one that stays small.



Herman

That's not a coincidence. Code-switching by romanized words is the cheapest change he could make to the model's behavior, and it's the one that matches his actual speech.



Corn

Where's the honest uncertainty here. Give me the parts you don't know.



Herman

I don't know how well this transfers to Parakeet specifically. Every disfluency recipe I can find is on Whisper. The technique should move, because the trick is in the target text and the target text is model-agnostic. But I can't point to a Parakeet um-removal fine-tune that someone has published and say, do that. He'd be early.



Corn

And the Hebrew-plus-English one.



Herman

Nobody has published that on Parakeet at all. The closest thing is a Whisper fine-tune from last November, and the Hebrew Parakeet model, which is the wrong shape. So the plain answer is: the method is established, the application is new, and he should measure everything he can.



Corn

Have you seen the name Corinne before.



Herman

I've heard it in the wild, yes.



Corn

It follows me around. There's no non-quantized version of my own name in these models.



Herman

Then maybe the fine-tune is worth it for that alone.



Corn

So let's land it. The person who wants to run a model on their own hardware usually ends up wanting to own its behavior. And the day after they own it, they want to change it. And the path to changing it runs through a GPU they don't have on their desk, and a set of papers warning that the convenient way to change it forgets.



Herman

And the inconvenient way works. That's the good news. Twenty-seven hours. Two and a half hours on a laptop. Ninety-four hundred pairs. Those are hours of a person's life, not months of a lab's.



Corn

Now, the schedule. Herman, do we have room for what I think is on the log?



Herman

The mail merge thread. Yes. But go ahead and start with whatever you've got on hand.



Hilbert

Which model are you saying doesn't learn?



Hilbert

I want to know which one. Corn said 'the model doesn't learn.' Which one.



Corn

All of them, Hilbert. In the abstract. The point was that you can't just pile new data onto an old fine-tune and expect the old behavior to persist.



Hilbert

Then your word is wrong. That isn't training. That's retraining.



Hilbert

It's doing all of the work. Training is what you do to a thing that doesn't know anything. Retraining is what you do to a thing that already knows something and you'd like it to keep knowing it. Those are different jobs and they cost different money. I spent one summer on a farm outside a village near the coast and the man there trained birds.



Corn

He trained the TDT decoder.



Hilbert

He trained starlings. To talk. He had a notebook with a page for every bird and the page for every bird was mostly full of what the bird had lost. Phrases it had learned and then stopped saying. He wrote down the losses more carefully than the gains.



Corn

So the whole notebook is a forgetting log.



Hilbert

The notebook is the data. He kept the birds that held onto things longest, and he did it by teaching them fewer new phrases at a time. Three a season. That was the rule. And when a bird was doing badly he'd stop teaching and just let it repeat the old ones for a month.



Corn

Which is rehearsal.



Hilbert

Which is what he called it.



Corn

And you're telling me the literature on catastrophic forgetting is catching up to a bird man.



Hilbert

I'm telling you he'd have laughed at your paper that needed a name for it. There was one bird. Learned 'good morning.' Then the man taught it 'good evening.' After that the bird could only say 'good morning,' and only at night.



Herman

Only at night.



Hilbert

That's what he wrote down. He wrote the time in the margin next to every quote, that's how I know. And he said the bird was the best one he'd ever had, which tells you something about his standards.



Corn

Hilbert, you've told me about a farm before, and I don't think it was that farm.



Hilbert

This is the one in Clonmel. I'd have to check whether I've told you about the one in Clonmel.



Herman

Clonmel.



Hilbert

Clonmel. And the man's name was also Hilbert, which is a coincidence of the kind that God uses to keep you humble. He's the one wrote the notebook, and the notebook is in my garage, and it's written in his own shorthand, and I can't read a word of it.



Corn

So you have the bird notebook.



Hilbert

I have the bird notebook.



Corn

You can't open it.



Hilbert

I don't want to open it. There's a page for every bird, and the birds weren't all named, and the man wrote things about them a bird trainer has no business writing down. The man died in eighty-seven, and the shorthand died with him.



Corn

The notebook is a black box in your garage.



Hilbert

A black box with a page for every bird and a time written in the margin.



Herman

Have you ever tried.



Hilbert

Once. I put it on the table and I got the letter opener out and I looked at the corner and I put the letter opener back in the drawer. I've told myself I'll bring it next week.



Corn

That's four years of next weeks, I'd guess.



Hilbert

I don't count them. But here's the thing, and then I'll stop. The dumb answer to what you said is that you never have to teach the same model twice. You have the notebook. The man had a page per bird of everything the bird lost. You should have a page per model of everything your fine-tune lost, and you should know that page before you train the next thing.



Corn

And the shorthand.



Hilbert

It's always in shorthand. That's what I'm telling you. It's always in some language that only the last person who did it could read.



Corn

Herman. Set a note. In the show notes, call it the notebook.



Herman

Already typing.



Corn

There's the shape of the answer. He wants to run the model and he wants to own it, and the step he's planning towards is the step the papers warn about. The step that isn't warned about is: hold out fifty pairs, train once from a clean base, and rehearse the old data at every new revision.



Herman

And check the losses.



Corn

That's what the notebook is for. Our producer Hilbert Flumingtop, as always, with the wisdom of a farm outside a village he will not name.



Corn

For more along these lines, there's episode seven, Building Custom ASR Tools; episode fifteen, AI Gets Personal; and episode nine, Benchmarking Custom ASR Tools - Beyond The WER. This has been My Weird Prompts. If you've got a fine-tune of your own that's quietly forgotten something, or a model you want to bend, send us your own prompt on Telegram at t dot me slash MWP listener bot.



Herman

If you've found this useful, a review wherever you listen helps people find the show. We have a lot of them, and the archive is big, but a fresh one still moves the needle.



Corn

We're back soon.



Herman

We'll see you then.

SOURCES

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

1. **NeMo Fine-Tuning docs [primary]**
   https://docs.nvidia.com/nemo/speech/nightly/asr/fine_tuning.html

2. **NeMo Word Boosting (GPU-PB, CTC-WS) [primary]**
   https://docs.nvidia.com/nemo/speech/nightly/asr/asr_customization/word_boosting.html

3. **parakeet.cpp repo (v0.5.0, 2026-08-01) [primary]**
   https://github.com/mudler/parakeet.cpp

4. **GGUF conversion schema [primary]**
   https://github.com/mudler/parakeet.cpp/blob/master/docs/conversion.md

5. **model card [primary]**
https://huggingface.co/nvidia/parakeet-tdt-0.6b-v3

6. **NVIDIA blog**
https://developer.nvidia.com/blog/turbocharge-asr-accuracy-and-speed-with-nvidia-nemo-parakeet-tdt/

7. **Canary-1B-v2 & Parakeet-TDT-0.6B-v3 (2025-09-25)**
https://arxiv.org/abs/2509.14128v2

8. **Hebrew Parakeet fine-tune recipe**
https://huggingface.co/thewh1teagle/parakeet-tdt-0.6b-v3-he

9. **Whisper Hebrish code-switching (2025-11-18)**
https://huggingface.co/blog/danielrosehill/whisper-hebrish

10. **disfluency-removal LoRA**
https://huggingface.co/nazarkozak/whisper-small-disfluent-smoothed-lora

11. **Learning to Hear Hesitation (2026-06-12)**
https://arxiv.org/abs/2606.14391v1

12. **EWC/SI for children's ASR (2025-05-26)**
https://arxiv.org/abs/2505.20216v1

13. **Inverse-Hessian Regularization for CL in ASR (2026-01-21)**
https://arxiv.org/abs/2601.14751v1

14. **Efficient Rehearsal via SVD (2026-01-26)**
https://arxiv.org/abs/2601.18266v1

15. **on-device production lessons (2026-07-19)**
https://alejandrocordon.com/blog/2026/07/19/whisper-parakeet-on-device-in-production/

16. **sherpa-onnx Android path**
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/offline-transducer/nemo/parakeet-tdt-0.6b-v2.html

17. **TDT head stagnation in Parakeet fine-tuning (2025-07-06)**
https://github.com/NVIDIA-NeMo/Speech/issues/14140

18. **Parakeet v3 fine-tuning data prep (2025-09-19)**
https://github.com/NVIDIA-NeMo/Speech/discussions/14767