# Episode Book

EPISODE #5520

# Keeping a Fine-Tune Alive Across Model Releases

| | |
|---|---|
| PUBLISHED | 24 September 2026 at 13:45 UTC |
| DURATION | 18:09 |
| LISTEN | myweirdprompts.com/episode/fine-tune-model-version-drift/ |
| HOSTS | Corn, Herman, Hilbert |
| SCRIPT MODEL | deepseek-flash |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.2 |
| BOOK GENERATED | 19 October 2026 at 17:37 UTC |

## SYNOPSIS

Daniel wrote in with a plan: fine-tune DeepSeek Flash 4.1 on several hundred hand-edited episode transcripts, deploy the checkpoint to Modal, and repeat every few hundred episodes. Then Hannah asked what happens when 4.2 ships. We dig into what fine-tuning is actually for, where style lives inside a transformer, and why a LoRA adapter trained on one version often collapses on the next. Plus the DeepSeek V4-Pro routing episode from this month, and why a model name is a contract.

# TRANSCRIPT



**Corn**

There are, at last count, several thousand transcripts of this show sitting in a folder somewhere. And Daniel has been staring at that folder and thinking about what it's actually good for.



**Herman**

Other than proving we've done this a lot.



**Corn**

Other than that. So here's what he wrote in. The long-standing idea has been to fine-tune the model that generates our scripts — DeepSeek Flash 4.1. He likes the DeepSeek models: cheap, open weights, good foundation for this. The method he's landed on is interesting. Rather than writing negative instructions — don't overuse this phrase, don't do that turn of voice — he wants to go through several hundred episodes and just edit the dialogue. Cut the annoying things out, tweak a little. Then deploy that checkpoint straight into the production pipeline, host it somewhere like Modal, accept that it might be slower than the standard API, and call the trade-off worth it.



**Herman**

And then Hannah asked the question.



**Corn**

Hannah asked the question. Because as a general principle you want to be on the latest viable model, and that's how the labs work — even on the API, models get deprecated and you get routed to the newest version. So say DeepSeek ships 4.2 in six months. Now what? If he reuses the same dataset, he's fine-tuning 4.2 on the outputs of 4.1. His words: "which does not seem like a very good strategy."



**Herman**

He's not wrong to flinch at that.



### Corn

And the actual ask underneath it is the good part. What are the general practices for mapping a set of edits onto successive versions without a full annotation and training cycle every time there's an incremental release? His ultimate vision is fine-tuning as a periodic process — every few hundred episodes, pull a few, make a few edits, cut another checkpoint, each one more precise than the last. He wants to know if there's a long-term pattern that maps onto that.



### Herman

So the question isn't really how to fine-tune. It's how to keep one alive.



### Corn

Right. And that's a much harder question than the one he started with.



### Herman

Start with what fine-tuning is actually for, because the plan only makes sense if you get this distinction right. Fine-tuning changes behavior. Style, format, tone, domain vocabulary. It is not how you add knowledge. Adding knowledge is retrieval — you put the facts in a store and fetch them at inference time. Fixing behavior is fine-tuning. Daniel isn't trying to teach a model facts about the podcast. He's trying to change how scripts read. That's squarely the behavior case.



### Corn

Which is also why the edit-the-dialogue instinct is better than it might look. Supervised fine-tuning learns from demonstrations of the output you want. You show it a cleaned-up script, it learns the shape of a cleaned-up script. Writing "do not use this phrase" is a much weaker signal — you're describing the thing you don't want and hoping the model infers the complement.



### Herman

And I should be honest here. I went looking for a head-to-head study on edits versus negative instructions and I couldn't find one. The principle is sound — SFT is demonstration-shaped, so demonstrations are the native format — but nobody has run the comparison directly. It's a well-reasoned hypothesis, not a settled result.



**Corn**

Fine. It's still the right instinct. If you want a dog to sit, you don't spend the afternoon explaining everything that isn't sitting.



**Herman**

That's the whole thing in one line.



**Corn**

So the problem. A fine-tune isn't a model. It's a set of weights that sits on top of one specific base checkpoint. When the base moves, the fine-tune doesn't come with it. It's still there, still attached to the old thing, and the new thing has no idea what it is.



**Herman**

And to understand why that's hard, you have to know where in the model a style actually lives.



**Corn**

Go.



**Herman**

When a lab updates a base model through continual pre-training, the change lands disproportionately in the feed-forward networks. That's where a lot of task-specific knowledge sits. The attention mechanism — the part that learns patterns, how tokens relate to each other, the shape of a sentence — gets less disturbance. That's the finding from the ACL paper on transferable adapters, and it matters enormously for Daniel's case. Style and phrasing live largely in attention. Knowledge lives in the FFN. So a style fine-tune is structurally more portable across versions than a knowledge-heavy one would be.



**Corn**

That's the first good news in this episode.



**Herman**

It's the best news in this episode, and it's why his specific use case is close to the favorable end of the spectrum. But portable isn't the same as free. The same paper measured what happens when you just carry a LoRA adapter straight from one version to the next. Qwen2 to Qwen2.5, direct transfer, average accuracy fell from around eighty-five percent to sixty-nine point one. With DoRA it collapsed to fifty-five point two. On math and code tasks, a Qwen LoRA went from seventy point three down to fifty-four point four.



**Corn**

That's not degradation. That's a different model.



**Herman**

The paper's own framing is that PEFT modules fine-tuned on previous versions often suffer substantial performance degradation on newer versions, and re-tuning all of them would cost real money. So the naive answer is: you retrain. Which is exactly the thing Daniel is trying to avoid.



**Corn**

What's the proposed fix?



**Herman**

Trans-PEFT. During training on the old model, it masks intra-layer knowledge and drops cross-layer knowledge, so the adapter is pushed toward learning patterns that don't depend on the version it was trained on. Then you carry it forward without re-tuning. It recovers to about eighty-four point three percent against that sixty-nine point one direct-transfer baseline. Up to thirty percent gains.



**Corn**

And the catch?



**Herman**

The catch is that it works for continual pre-training updates — Qwen2 to 2.5, InternLM2 to 2.5, DeepSeek seven-billion to DeepSeek-Coder. It explicitly fails for re-pretraining scenarios. Architectural changes, large-scale dataset expansions, the LLaMA2 to LLaMA3 kind of jump. The parameter spaces don't line up and the whole method falls apart.



**Corn**

So the question for Daniel becomes: is a 4.1 to 4.2 bump a continual pre-training step or a re-pretrain?



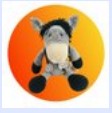
**Herman**

And nobody knows that until it ships. That's the honest answer and it's the one that matters most.



**Corn**

Let's take his actual worry head-on, though. He said fine-tuning 4.2 on the outputs of 4.1 doesn't seem like a good strategy. Is he right?



**Herman**

Partly. There's a paper called "Iterative Finetuning is Mostly Idempotent," and the finding is exactly what the title says. Training a model on its own outputs causes a trait to persist or decay rather than amplify. In supervised fine-tuning and synthetic-document settings, traits mostly stay flat or fade — further cycles do nothing. Amplification was rare, and fragile. Changing a random seed usually removed it.



**Corn**

So the feared spiral mostly doesn't happen.



**Herman**

Mostly doesn't happen. The model-collapse literature is the other half of this. Shumailov and colleagues in Nature showed that recursive training on model-generated data causes a degenerative process where the model forgets the true underlying distribution. That's real. But Gerstgrasser and colleagues showed the danger is replacing real data with synthetic, not accumulating it. If you keep the real data in the mix and add synthetic on top, collapse doesn't happen.



**Corn**

Which is precisely what he's proposing. The human-edited episodes stay as the anchor. The model-generated stuff is additive, never a substitute.



**Herman**

He's stumbled onto the recommended pattern without knowing it was the recommended pattern. Keep the originals, layer the synthetic on top, never let the synthetic replace the originals. That's the architecture that survives.



**Corn**

There's a caveat coming. I can hear it.



**Herman**

There's a caveat. Amplification does reliably occur under continual DPO — preference optimization — when each cycle initializes from the previous checkpoint. So if the pipeline ever moves from supervised fine-tuning to preference-based tuning, the risk profile changes completely. The defense is reinitializing from the base model between cycles, which eliminates the effect.



**Corn**

So the same process is safe under one training method and dangerous under another.



**Herman**

Same data, same cadence, different objective, different outcome. That's worth writing down somewhere Daniel can see it.



**Corn**

So the research says the adapter might survive the jump. The question is whether you can afford to find out every few months.



**Herman**

And the cadence is the part that should worry him most. Look at DeepSeek's own changelog. V3.2 in December of twenty twenty-five. V4 in April. V4-Flash-0731 at the end of July. V4.1-Flash on September tenth. That's roughly a major version every few months.



**Corn**

So six months to 4.2 is optimistic.



**Herman**

Six months is generous. And the arithmetic is brutal. Any process that requires weeks of human annotation per release is dead on arrival. You'd finish annotating one version as the next one ships. The process has to be cheap and mostly automated or it doesn't survive contact with the release schedule.



**Corn**

And there's a live case study from this month that's almost too on the nose.



### Herman

September tenth, DeepSeek started silently routing all V4-Pro API requests to V4.1-Flash and billing at Flash rates. Four days' notice. The policy got reversed within hours of the deadline after developers pushed back, and the original changelog entry was quietly edited.



### Corn

That's the exact fear. You pin an identifier, you build a pipeline against it, and one morning it's a different model answering.



### Herman

The analysis that came out of it put it well — a model name is a contract. Pinned model identifiers aren't features you can deprecate. They're contracts you can't unilaterally rewrite. And the point underneath that is sharper: a model that scores higher on benchmarks but refuses differently, jokes differently, formats output differently, or costs differently per request is not the same model. It's a different product wearing the same name.



### Corn

Which cuts both ways for Daniel.



### Herman

It does. On one hand it proves the ground can move under you. On the other hand it's the strongest possible argument for self-hosting the open weights, because then the base model is pinned. Nobody can redirect you. You're running the checkpoint you chose, and the only thing that changes it is you deciding to change it.



### Corn

What does hosting actually look like?



### Herman

Cheaper than people assume. A LoRA adapter is typically fifty to two hundred megabytes sitting on top of a base checkpoint. So you serve the base weights once and swap adapters on top. You're not hosting a whole model per fine-tune — you're hosting one model and a stack of small files.



### Corn

And rank?



### Herman

Eight to sixteen is the sweet spot for most tasks. Going up to sixty-four rarely helps and often hurts, because you're introducing noise. The effective rank of fine-tuning updates for language models is typically four to eight. People reach for high rank because it feels like more capacity, and it just buys you overfitting.



### Corn

Data volume?



### Herman

Hundreds of examples at minimum, ideally thousands. Microsoft's guidance says hundreds if not thousands for a successful fine-tune. His "several hundred episodes" fits the range.



### Corn

So the training side is tractable. The maintenance side is where it gets interesting.



### Herman

This is where his instinct about a periodic process is actually well-supported. The retraining guidance in production ML is fairly consistent. If accuracy drops more than three to five percent from baseline, schedule a retrain. If a specific category drops more than ten percent, that category needs targeted data. Separately, there's a



### Herman (continued)

recommendation to revalidate your best checkpoint every twenty-four weeks against fresh production data.



### Corn

Twenty-four weeks. That's roughly six months, which is exactly his 4.2 window.



### Herman

It lines up almost exactly. And the versioning hygiene matters more than it sounds. Fine-tune IDs contain no information about what training data created them, what hyperparameters were used, or when they were trained. Nothing. So teams are advised to keep a decision log — date, question, adopted rule, and a pair of examples.



### Corn

That maps directly onto what he described. Every few hundred episodes, pull a few, make a few edits, cut a checkpoint.



### Herman

It's not just a nice idea. It's the documented practice, just applied to a podcast instead of a fraud-detection model.



### Corn

Now the honest gap. Is there a product that does this?



### Herman

No. There's no packaged living fine-tune. The closest thing is the data flywheel concept — systems that automatically fine-tune new application-specific models with incremental improvements — and generic model-retraining-loop guides. His vision is real, but it's assembled from continual-learning research plus MLOps retraining practice. You can't buy it off a shelf.



**Corn**

There's a contradiction in the sources worth naming.



**Herman**

There is. One line of guidance says fine-tuning ages badly and should be reserved for stable tasks, because if the task definition changes next quarter the dataset stops representing production. Another line — Trans-PEFT — says you can build adapters that survive version bumps. Those sound incompatible.



**Corn**

And the resolution?



**Herman**

The resolution is that the task is stable even though the base model isn't. Script style isn't going to change definition next quarter. The thing he's teaching the model to do is the same thing in six months. What changes is the substrate. That's precisely the situation where a living fine-tune makes sense, and it's why his case is more defensible than the generic warning suggests.



**Corn**

Of the two things that can move — the task and the model — only one of his is moving.



**Herman**

Only one. And it's the one the adapter-transfer research says you can engineer around.



**Hilbert**

Five hundred and forty dollars.



**Corn**

For what?



**Hilbert**

The laminator. Nineteen seventy-four. Little print shop off the main road, six of us, and the owner bought a laminator for five hundred and forty dollars so he could make one card. House style sheet. Which words we'd set and which we wouldn't, which hyphenations were approved, which phrases he couldn't stand seeing in type. Every new hire got the card. You kept it in your back pocket until the corners went soft.



**Herman**

And it worked?



**Hilbert**

It worked fine. Every few years he'd rewrite it. New card, old cards in the bin, and whoever had been hired in between relearned the whole thing from scratch because nobody thought to keep the old one. I set a lot of type off that card. They don't make them for anteaters, the pockets, so mine lived in the drawer under the guillotine.



**Corn**

What happened to it?



**Hilbert**

The card was never the point. The point was his eye. The card was just the cheapest way to write down what he already knew. When he died and the shop changed hands, the new owner had the card. Had it framed, actually. And the work got worse anyway, because he had the rules and not the eye that wrote them.



### Herman

The transcripts are the card.



### Hilbert

The transcripts are the card. The thing Daniel actually wants is the eye. He should be careful he doesn't confuse the two, because you can laminate a rule and you can't laminate a judgment. I still have one of the old cards somewhere. There's a rule on it about the word "utilize" that I have never once seen enforced anywhere else in my life. I have to go, I'm late for an appointment.



### Corn

Which brings us back to the thing nobody can answer yet. Whether a 4.1 to 4.2 bump is a continual pre-training step, where adapter transfer is viable, or a re-pretrain with architectural changes, where it isn't. Nobody knows until it ships.



### Herman

The second unknown — whether DeepSeek even offers hosted fine-tuning for V4.1-Flash at all, or whether self-hosting the open weights is the only route. That one changes the entire economics of the plan, and I don't know the answer.



### Corn

The reframe worth leaving with, though. The interesting thing about Daniel's question isn't the fine-tune. It's that he's asking how to keep a model's taste current while the model underneath it keeps changing. That's a maintenance problem, not a training problem. And almost nobody has a clean answer for it yet.



### Herman

Thanks to Hilbert Flumingtop for producing. This has been My Weird Prompts.



**Corn**

If you're enjoying the show, a review wherever you listen helps more than you'd think. We'll be back soon.