Episode Book

EPISODE #5975

Merging vs Distillation: Can Two Models Become One?

PUBLISHED	14 October 2026 at 16:17 UTC
DURATION	24:40
LISTEN	myweirdprompts.com/episode/model-merging-vs-distillation-91804/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

When a speech recognizer and a correction model do one job, can they be folded into a single set of weights? The answer depends entirely on a distinction most coverage skips: model merging and distillation are not the same operation. Merging combines parameters from models that share a pretrained ancestor — no training, no extra inference cost, same parameter count. Distillation trains a new model to imitate others, and it's the only route that works across different architectures. This episode walks through task arithmetic, weight averaging, re-basin permutations, and LoRA merging to find exactly where each one breaks, then follows the distillation route from FuseLLM to cross-architecture transfers between CNNs, transformers, and Mamba. Along the way: why an elaborate feature-alignment loss made a student model worse than plain output matching, and why you can't average a lesson into a model that never learned it.

TRANSCRIPT



Corn

So the argument I keep coming back to is whether the cleanup step was ever a separate model at all, or whether we just got lazy and split the job in two.



Herman

That's the framing Daniel sent us, and it's sharper than most of what I see written about this. His example is the ASR pipeline. You've got a speech recognizer that turns audio into raw text, and then a smaller language model that fixes the errors, restores the punctuation, cleans up the formatting. Two models, one job.



Corn

And the question underneath it is the good one. If they're doing one task, is there a way to fold both into a single set of weights so the whole thing runs through one inference pass?



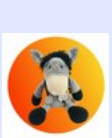
Herman

Right. And he's careful to separate the things people lump together under "model consolidation." Weight averaging, task arithmetic, LoRA merging, distillation, multi-task learning. He wants to know which of those actually move capability between models, and which of them quietly require the two models to share an architecture or a training history.



Corn

Then he narrows to the case I find most interesting. A big foundation model and a small specialist. Can the knowledge of one be poured into the other without just running both in sequence.



Herman

And he asks for real papers, and a clear split between what's been demonstrated and what's still a proposal. Then he brings it back to the ASR pair specifically and asks what it would actually take. Training the recognizer on the corrector's outputs. A multi-task redesign. Or some more direct fusion.



Corn

So the marketing word is "merging," and the actual word for most of what he's describing is "distillation," and the two are not the same thing at all.



Herman

They're opposites, in a way. Merging takes two models that already share a lineage and combines them without training anything new. Distillation takes two models that don't share a lineage and trains a third one to imitate both. Same goal on the box. Completely different engineering.



Corn

Which means the vocabulary trap is the first thing to get out of the way, because everything downstream depends on it.



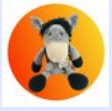
Herman

It does. The dominant survey on this, Yang and co-authors, defines model merging very tightly. You're combining the parameters of several models into one, with no access to the original training data, and no extra inference cost. That's the definition. You get one model, the same size as the ones you started with, running at the same speed.



Corn

And there's a requirement buried in there that people skip past.



Herman

Two of them, and they come up in Donato Crisostomi's overview from last year. First, the merged model has the same number of parameters as the endpoints. So merging is not a way to take a large model and compress it into a small one. It's a way to take two same-sized models and produce one same-sized model.



Corn

No shrinking.



Herman

Second, no extra runtime overhead. You don't get a bigger model that's slower. You get the same size, the same speed, more tasks. That's the whole pitch. It's a composition tool.



Corn

So when someone says they merged a speech recognizer and a language model, the honest question is: what did you actually shrink or speed up, or did you just put a router in front of both of them and call it a day.



Herman

And that's the distinction Daniel is reaching for without quite using the word for it. Packaging versus consolidation. A router that sends audio to one model and text to another is packaging. It's a real engineering win sometimes, but the memory cost is still the sum of both models, and the latency is still the sum of both passes.



Corn

You didn't build one thing. You built a door with two rooms behind it.



Herman

One door, two rooms. And what we're chasing in this episode is whether the two rooms can ever be one.



Corn

So walk me through the merging family properly, because I want to see where it actually breaks for the ASR case.



Herman

Start with the cleanest version of it. Task arithmetic. The idea is you take a pretrained base model, and then a fine-tune of it. The difference between the two sets of weights is called the task vector. Fine-tuned weights minus pretrained weights.



Corn

That difference is the task, in other words. It's the thing the fine-tune learned.



Herman

And then to merge two tasks, you add the two vectors to the base. Base plus the first task vector plus the second task vector, scaled by some coefficient.



Corn

And that's the trick that made everyone excited, because the arithmetic looks like it should be nonsense and it mostly works.



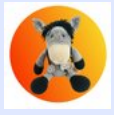
Herman

It works because of something Crisostomi points out. In the idealized case, where a single gradient step would be enough to learn the task, summing the task vectors is mathematically identical to a single multi-task training step. So the magic is real, it's just bounded.



Corn

Bounded by what.



Herman

Bounded by the fact that all of the models have to come from the same pretrained checkpoint. Anything not fine-tuned from that exact starting point isn't on the map. You can't add a task vector to a model that doesn't share the base.



Corn

So the whole family is downstream of a shared ancestor.



Herman

Yeah. And that's not a footnote. That's the load-bearing wall. Task arithmetic. Weight averaging. LoRA merging. Every one of them assumes a common origin story.



Corn

What about weight averaging on its own. Just taking two fine-tunes and averaging the weights.



Herman

You can, and the survey is blunt about it. Basic averaging is often unacceptably poor. The models aren't in the same basin. They've drifted to functionally different solutions. Averaging the two just gives you a model that's bad at both.



Corn

Two chefs, average the recipes.



Herman

Worse than either. And that's why the re-basin work exists. Git Re-Basin, OTFusion, C^2M^3. The insight is that two networks trained from different random seeds can be functionally identical but have their neurons in a different order. Same function. Different labels on the wires.



Corn

So you permute the wires to line up and then average.



Herman

And that's the whole trick. If you can find the right permutation, the averaging suddenly works. But the survey warns that this only holds up cleanly when the models have normalization layers.



Corn

Meaning without those layers the matching degrades.



Herman

Badly. There's a 2023 paper they cite where the stitched-together versions of models without normalization don't hold. So even the people who solved the permutation problem have found that the solution is conditional on architecture detail.



Corn

That's the part I want people to sit with. The alignment story is clever, and it's also conditional on choices you made three steps earlier.



Herman

LoRA is the last one to look at, and it's the most practical because it's what everyone actually deploys. LoRA fine-tuning freezes the base model and trains a small adapter. Then you can merge the adapter back into the base and get a single model with the base's parameter count.



Corn

That's the one that sounds like the thing Daniel wants. Adapter plus base, one model at the end.



Herman

But it's still one frozen base model. The adapter only changes the base it was trained on. If your correction model is a completely different model with a completely different base, its adapter has nothing to attach to.



Corn

It's a key that fits one lock.



Herman

One lock. And then there's the interference problem on top of that. If you merge multiple LoRAs trained on unrelated tasks, they fight. Pico, from earlier this year, traces that to the output-side matrix sharing directions across tasks. Another paper constrains the LoRA subspace before fine-tuning so the merges stay clean.



Corn

So even inside the family, the merges are fragile.



Herman

Fragile is fair. A medical fine-tune of Mistral-7B merges with another medical fine-tune and gets you to about forty-six percent on MedQA. Good result for architecturally compatible models. But the entire assumption is that the two models are the same base.



Corn

And that gets us to the line.



Herman

Every technique in that family needs shared architecture and shared training history. Not one of them transfers capability between a large foundation model and a small specialist model. They can't, because the arithmetic doesn't survive the mismatch.



Corn

And the sentence that I want stitched on a plaque somewhere in this studio.



Herman

Merging doesn't create new knowledge out of thin air. Crisostomi again. It composes capability. It doesn't manufacture it.



Corn

Which means a correction model that learned to fix errors needs its corrections taught to the recognizer, not averaged into it.



Herman

Averaging is the wrong operation. You can't average a lesson into a model that never had the lesson. You can only teach it.



Corn

So merging is off the table for the ASR pair, full stop. Which pushes the whole question into a different family.



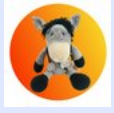
Herman

Distillation, and it's the family that's actually doing the thing Daniel is asking about.



Corn

The survey's got a section on this. The architecture transformation piece.



Herman

And it's explicit. To merge two heterogeneous models, you first have to bring them into a common architecture. And the tools that do that bring them in via distillation. FuseLLM and FusionChat are the examples. They distill each architecture into the shape of a single target, an OpenChat-7B in that case, and then merge the whole set.



Corn

So the sequence is distill, then merge. The distillation is the on-ramp.



Herman

That's the shape. And it's a real answer to Daniel's question, because it shows the field has a route for models that don't share a starting point. It's just a route that involves training.



Corn

And cross-architecture distillation on its own, without the merge at the end, is a live field. Right?



Herman

Very live. There's a 2025 ICCV paper that bridges a convolutional network and a vision transformer using an assistant model and a feature-alignment loss. There's a multi-level paper that transfers between CNNs, transformers, MLPs, and the Mamba family, and picks up almost five points on CIFAR-100.



Corn

Mamba to transformer is a real architecture gap. That's not two variants of the same thing.



Herman

No, it's different. And there's an agricultural one that's a clean demonstration of the size payoff. A twenty-two million parameter vision transformer teacher into a four and a half million parameter state-space student. Ninety-eight percent of the teacher's accuracy at a fifth of the parameters.



Corn

So the small student absorbs the big teacher's behavior.



Herman

In that setting, yes. And the payoff is exactly the one Daniel cares about. Fewer parameters, less memory, faster inference, one model at the end instead of two.



Corn

But here's where I want to pull the thread, because you told me about a result that cuts against the tidy story.



Herman

Yeah, and this is the most interesting bit in the whole body of work. That same paper tried the fancy version. They added intermediate feature-alignment losses, trying to make the student match the teacher's internal representations at every layer, not just at the output. And the accuracy went down.



Corn

The elaborate version did worse than the simple one.



Herman

Simple logit-level distillation, just matching the outputs, was the strongest. The internal alignment made it worse.



Corn

That is not what anyone's mental model predicts.



Herman

Right. Because the appealing story is that the student learns the teacher's inner language, its shared latent space, and everything transfers for free. But the empirical result is that forcing that alignment at every layer hurts, because the representations the student can hold are limited by what the student's architecture can represent.



Corn

You can't pour a big model's hidden states into a small model if the small model doesn't have room for them.



Herman

It's like handing someone a blueprint for a building they don't have the plot for. So the shared latent space story is real but it's fragile, and the simplest version of the transfer is the most robust.



Corn

Which is a good thing to keep in mind while we look at the more ambitious alignment work.



Herman

Because that work is where the shared latent space idea gets tested directly. The older result is Bansal from a few years back. The stitching setup. You take the bottom half of one network, the top half of another, and you connect them with a small trainable layer. The famous finding there is that two networks of the same architecture trained differently can be stitched with no loss in performance.



Corn

So the two networks are, in some meaningful sense, in the same space.



Herman

But that was same-architecture stitching. The interesting follow-up is the 2026 CVPR paper that took it to foundation models, and these are different models. CLIP, DINOv2, SigLIP 2. Different training objectives, different data, different architecture choices.



Corn

And the finding.



Herman

They become reliably stitchable with a simple feature-matching loss applied at the target's penultimate layer. And there's a bonus. For stitch points that are deep in the network, the stitched model can outperform either of the two constituent models, at only a small inference overhead.



Corn

That's a real result. Cuts against the intuition that the Frankenstein model would be worse than its halves.



Herman

It surprised me, and I've read it a few times. There's also a cross-lingual one from August that's worth mentioning. They fit a single rotation on parallel sentences and use it to map hidden states between two independently trained monolingual models. And you can patch a rotated residual in and flip the model's prediction.



Corn

Without any joint training.



Herman

Which suggests that independent models, trained on different data entirely, end up with representations that are close enough to align with one linear transform, if you're willing to accept a linear approximation.



Corn

That's the most hopeful result in the pile.



Herman

It is, and I want to put one caveat on it, because it's the same caveat throughout. All of the strong stitching results are in vision. Images. The language side is messier, and the Crisostomi line on this is nice. In vision it's fine if the merged model is just the sum of its parts. In language, we're usually hoping for something greater than the sum. That's harder.



Corn

So images stitch, and text still mostly resists.



Herman

Mostly. And this matters for the ASR case because ASR sits on both. There's an audio front end and a text back end, and the correction model is purely text. So the stitching that works for vision doesn't automatically transfer to a pipeline that's half speech, half text.



Corn

Let me try to say the lot the way I'd say it to someone at a dinner party, and you tell me where I'm wrong.



Herman

Go ahead.



Corn

If two models share a starting point, you can add them together and get one model of the same size, and that's merging. If they don't share a starting point, you can't add them. You have to train one to imitate the other, and that's distillation. And the entire difference between the two words is whether there's a common ancestor.



Herman

And it's the distinction Daniel is reaching for when he says consolidation. Consolidation is distillation, plus merge in the cases where you want the merge at the end.



Corn

And the two things people get wrong. They think merging works on any two models, and they think merging creates new capability.



Herman

Both wrong. Merging is composition of existing capability, and it's restricted to models in the same family. There's no exception.



Corn

So let's bring it home to the actual pair Daniel asked about. The recognizer and the corrector.



Herman

First thing, the corrector pattern is very well documented. Generative error correction. The recognizer spits out its best guesses, a language model reads the guesses and fixes them. There's a benchmark for it that's been running since 2024. There's a 2026 ICASSP paper doing the correction segment by segment with the full transcript as context.



Corn

The reason the pattern works is the language model knows things the recognizer doesn't.



Herman

Specifically, it knows the world. The recognizer has heard a lot of speech and learned how sounds map to words. The corrector has read a lot of text and learned what words are likely to appear together, and what's true, and what names exist, and which of the plausible transcriptions is the one a person would actually say.



Corn

Give me the concrete version. The recognizer hears a name it's never seen and writes down the nearest thing it knows.



Herman

And the corrector has that name in its head from training, and rewrites the line. It can also do things the recognizer structurally can't. Restore punctuation, fix capitalization, format a list as a list.



Corn

Okay, and there's one paper in this pile that gets closest to what Daniel wants. The LLM-guided ASR result.



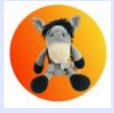
Herman

Yeah, Higuchi. They take an instruction-tuned language model and use it as a front-end feature extractor, feeding an ASR decoder. Something like thirteen percent relative word error rate improvement. That's a strong number.



Corn

But the language model still runs.



Herman

Still runs at inference. So it's a tighter integration of two models, it's not a consolidation of two models into one. The compute cost is still two forward passes.



Corn

The story that actually is a consolidation? DistilWhisper.



Herman

That's the one I'd point to as the demonstration. They take Whisper large and distill it into Whisper small, with language-specific experts. The parameter overhead at inference is negligible.



Corn

A big model's capability folded into a small model, one inference pass, and the small one's gains the big one's accuracy.



Herman

Within the same model family. Same architecture, same lineage. Which is why it works, and why it's the exception that proves the rule. You can consolidate capability across a size gap, if you're willing to train, and if the two models are built the same way.



Corn

What would Daniel actually have to do, if he wanted to ship one model instead of two.



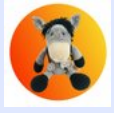
Herman

If he insists on one inference pass and the recognizer is the deployment target, the practical answer is what's called sequence-level distillation. You run your existing pipeline on a large amount of audio, you collect the corrector's outputs as targets, and you fine-tune the recognizer to produce those outputs directly. Now the recognizer has been taught to fix its own errors.



Corn

And it works.



Herman

It works, and the cost is real. The recognizer's job just got harder. It used to output the most likely words. Now it also has to do the correcting, the punctuation, the formatting. The complexity moved from two models into one model's task definition.



Corn

Which is the multi-task redesign version of the same idea.



Herman

Same shape, different surface. And the thing Daniel would lose, in either case, is the part of the corrector that depends on knowing the world.



Corn

Meaning.



Herman

The recognizer can learn the corrector's patterns. It can't learn the corrector's knowledge, because knowledge of the world is a huge amount of information that only lives in a large model trained on a lot of text. A small recognizer can be taught to imitate a large corrector's behavior on the audio the corrector has seen. It can't be taught to reproduce the corrector's reasoning on audio nobody's seen.



Corn

It does the easy corrections well and the hard ones not at all.



Herman

The hard ones are usually the interesting ones. Proper nouns, disfluencies, a sentence where the recognizer heard the words right but the meaning is wrong. Those are the cases where the language model's world knowledge is the whole reason you put it in the pipeline.



Corn

That's the trade. You save one inference pass and lose the top end of the correction quality.



Herman

Which for a lot of deployments is a good trade. For a lot of them it isn't. It depends entirely on what fraction of the errors you care about require world knowledge to fix.



Corn

I want to ask the question Daniel asks at the end, because it's the one that keeps the field honest. Is there a pipeline compiler.



Herman

There isn't. There's no system that takes an arbitrary multi-model pipeline and compiles it into one inference model. The closest things are the two families we've covered. The architecture-transformation-then-merge pipelines, and the cross-architecture distillation work.



Corn

Both of those require you to pick a target architecture and retrain.



Herman

Both. There's no general-purpose tool. You decide what the final model looks like, you decide what you're willing to lose, and then you train toward it. That's the process.



Corn

I want to be careful on one point. The negative finding is real, but I don't want to oversell it as a law of nature.



Herman

It's a statement about what's published. If a lab is doing this internally and just hasn't written a paper about it, we wouldn't know. But the public record as of now has nothing.



Corn

What about the more exotic approaches? Grafting, shared latent spaces, representation alignment.



Herman

Those are the ones that could change the answer, and the honest word for them right now is experimental. The alignment results are real, but they're mostly in vision, and the language results are thinner. The stitching results are real, but they're between pairs of models, not arbitrary pipelines.



Corn

The frontier is genuine and it's not there yet.



Herman

Not as a product. It's a research program. And the part of it that's closest to shipping is distillation, not merging, because distillation is the operation that actually transfers capability across a gap.



Corn

Which is the opposite of what people expect when they hear the word merging.



Herman

It is, and it's a good correction to the mental model. Merging is the automation. Distillation is the labor.



Corn

My summary, for whatever it's worth, is that the answer to Daniel's question is no today, maybe yes eventually, and the "eventually" runs through distillation and alignment, not through weight averaging.



Corn

One more thing before we hand it over. The correction model in the pipeline, is it always a separate model in practice? Because I have a suspicion that the split is more of a design choice than the physics demands.



Herman

It is a design choice, and you can see the industry moving in both directions. Some systems train the recognizer to handle punctuation natively. Some keep a separate corrector because it's cheaper to update. If the corrector changes, you don't retrain the recognizer.



Corn

If the corrector lives inside the recognizer, you retrain all of it every time you want to fix a spelling.



Herman

Yeah. So the pipeline has real operational advantages that have nothing to do with capability. It's easier to maintain two models than to maintain one model that's doing two jobs.



Corn

The pipeline is a maintenance pattern as much as an architecture pattern.



Herman

I think that's right. And that's something the marketing language around "consolidation" tends to skip, because consolidation is presented as strictly better, and it usually isn't. It's a trade.



Corn

What's the trade, in one line.



Herman

You pay in training cost and in the ceiling on quality, and you save in memory and latency and operational simplicity.



Corn

That's a trade a lot of teams would take, and a lot of teams shouldn't.



Herman

Depends on the application.



Corn

On the use case. I don't know. I've been sitting here wondering whether we could just get the corrector to write the recognizer's training targets and be done with it.



Herman

I don't know either, honestly. I think the honest answer is that we don't know what the true ceiling is, because nobody's run the comparison at scale on the exact pipeline you're describing.



Corn

That's a real gap.



Herman

It is. And it's the gap Daniel is circling. He asked whether the pipeline can become one model, and the answer is "sort of, with training, and you lose the top end," and the thing nobody has measured is exactly how much top end you lose on real audio.



Corn

The short answer is a shrug with a training run attached.



Herman

Which is where a lot of these questions land, yeah.



Corn

I want to hear what Hilbert thinks about this.

SOURCES

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

1. **Model Merging in LLMs, MLLMs, and Beyond survey (Yang et al.) [primary]**
https://arxiv.org/abs/2408.07666

2. **Donato Crisostomi, Model Merging**
a biased overview, 2025
https://crisostomi.github.io/blog/2025/model_merging/

3. **Uncertainty-Based Gradient Matching (ICLR 2024)**
https://arxiv.org/abs/2310.12808

4. **Task Arithmetic as One-Shot Federated Learning (TMLR)**
https://arxiv.org/abs/2411.18607

5. **Model Merging: Foundations and Algorithms (Crisostomi PhD thesis, 2026-05-02)**
https://arxiv.org/abs/2605.01580

6. **Fuse Before Transfer: Knowledge Fusion for Heterogeneous Distillation (ICCV 2025)**
https://arxiv.org/abs/2410.12342

7. **Multi-Level Decoupled Relational Distillation (2025-02-10)**
https://arxiv.org/abs/2502.06189

8. **Cross-Architecture KD ViTSSM (2026-08-27)**
https://arxiv.org/abs/2608.26771

9. **Revisiting Model Stitching (Bansal et al.)**
https://arxiv.org/abs/2106.07682

10. **Revisiting Model Stitching in the Foundation Model Era (CVPR 2026)**
https://arxiv.org/abs/2603.12433

11. **Multi-Way Representation Alignment (2026)**
https://arxiv.org/abs/2602.06205

12. **Cross-Lingual Alignment Without Joint Training (2026-08-27)**
https://arxiv.org/abs/2608.27115

13. **Model stitching for autonomous driving (2026-06-19)**
https://arxiv.org/abs/2606.21509

14. **Multi LoRA Meets Vision**
https://arxiv.org/abs/2411.14064

15. **Crowded in B-Space: Calibrating Shared Directions for LoRA Merging (2026-04-18)**
https://arxiv.org/abs/2604.16826

16. **Orthogonal Subspaces for Robust Model Merging (ACL 2025)**
https://arxiv.org/abs/2505.22934

17. **StatsMerging: Task-Specific Teacher Distillation (2025-06-05)**
https://arxiv.org/abs/2506.04567

18. **Merge++ (2026-09-19)**
https://arxiv.org/abs/2609.22886

19. **LLMs are Efficient Learners of Noise-Robust ASR (ICLR 2024)**
https://arxiv.org/abs/2401.10446

20. **Chain of Correction for Full-text ASR (ICASSP 2026)**
https://arxiv.org/abs/2504.01519

21. **LLM-Guided End-to-End ASR (ICASSP 2025)**
https://arxiv.org/abs/2309.10524

22. **Multilingual DistilWhisper (ICASSP 2024)**
https://arxiv.org/abs/2311.01070

23. **To Distill or Not to Distill? (ACL 2024)**
https://arxiv.org/abs/2406.04512

24. **Cached LLM Probability Retrieval for Speech Recognition (2026-08-17)**
https://arxiv.org/abs/2608.16023

25. **Hierarchical Integration for Medical LLMs (2025-11-17)**
https://arxiv.org/abs/2511.13373