Episode Book

EPISODE #6693

Why Fine-Tuning Doesn't Change a Model's File Size

PUBLISHED	19 November 2026 at 22:18 UTC
DURATION	31:12
LISTEN	myweirdprompts.com/episode/fine-tuning-model-file-size-74193/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	20 November 2026 at 03:35 UTC

SYNOPSIS

A fine-tuned Parakeet TDT 0.6B v2 speech model stays roughly 660 megabytes before and after training, yet its accuracy noticeably improves. Because a safetensors file is a fixed grid of allocated numbers rather than an empty warehouse, fine-tuning only nudges existing values in place. The improvement lives in the difference between the base and fine-tuned weights—a low-rank task vector—which can be inspected directly on a CPU using tools like wdiff.

TRANSCRIPT



Corn

A file that refuses to change size but plainly changes behavior is the kind of thing that makes you distrust your own eyes. Daniel's back on his Parakeet fine-tune. Same safetensors file, roughly six hundred and sixty megabytes before he trained it, roughly six hundred and sixty after, and yet the model unmistakably got better at recognising his speech. So he's asking three things at once. Where did the improvement physically go? Can he actually see it — can he compare the before and after checkpoints and find what changed and where? And the big one underneath: how does a fixed array of numbers hold more knowledge just because the values got nudged around?



Herman

And the short answer to the first part is that the file never had room to grow. Which, when you sit with it, tells you something about what a model file even is.



Corn

Say more about that, because I think people imagine the file as a warehouse. You train the model, the warehouse fills up a bit more.



Herman

Right, and the warehouse picture fails on the very first inspection. A warehouse has empty shelf space waiting to be filled — that's what makes it a warehouse. A model file has no empty space at all. Every byte is allocated before training begins. There is no "room" in the file because the file is not a room. It's a fixed grid of slots, every slot already holding a number, and training's only available move is to change what's in the slots.



Corn

The size question dissolves the moment you do the arithmetic, so let's do the arithmetic. Parakeet TDT 0.6B v2, the one Daniel fine-tuned, is six hundred million parameters by architecture. That's on NVIDIA's model card. Six hundred million numbers, each stored at roughly one point one bytes in sixteen-bit precision with a bit of header overhead, and you land at about six hundred and sixty megabytes. Fine-tuning doesn't add layers, doesn't widen a single matrix, doesn't grow the vocabulary. The byte count isn't just stable — it's fixed by the architecture before a single training step runs.



Herman

And it's worth stressing the vocabulary point, because that's the one place a speech model could plausibly grow. If fine-tuning added a single new token to the tokenizer, you'd need a new row in the embedding matrix and a new column in the unembedding, and the file would change size — by kilobytes, but it would move. Daniel's recipe doesn't do that. Same alphabet of sounds, same output space, same matrices. The architecture is a closed set of slots.



Corn

The reframe that helped me, and I think it's the one to carry through the whole episode, is that the file is a photograph, not a container. Fine-tuning doesn't add pixels to a photograph. It re-exposes it. Same dimensions, same byte count, different picture.



Herman

Which means the improvement isn't stored anywhere in the file. It is the difference between the two files, and a difference doesn't exist as a thing until you subtract.



Corn

That's the sentence I had to read twice. A difference doesn't exist until you subtract.



Herman

It's a bit of a logical sleight of hand that we're used to in other domains and find eerie here. If you take a photo of your kitchen, rearrange the kitchen, and take another photo, the "rearrangement" isn't printed on either photograph. It's only visible when you put the two side by side and compare. The change lives in the comparison, not in either artifact. Daniel is looking at one photograph and asking where the rearrangement is printed. It isn't printed. It's between.



Corn

There's one wrinkle in Daniel's own setup worth flagging before we chase it. His recipe is NeMo, and we don't know — I don't — whether his run was a full fine-tune or a LoRA adapter that got merged back into the base weights. Both produce a roughly six-hundred-sixty-megabyte file at the end. But a merged adapter's delta is low-rank by construction, because that's the whole design, and a full fine-tune's delta is only low-rank empirically, as a finding rather than a guarantee. Same file size, different diff story.



Herman

So step one for Daniel is opening the config YAML and seeing which flag got ticked.



Corn

Step zero. The mechanics first. A safetensors file is a small header — tensor names, dtypes, shapes, byte offsets — and then a flat blob of raw tensor bytes. We covered the container itself back when we did the episode on safetensors versus the other inference formats, so I'll spare you the tour. The point today is that fine-tuning changes none of the header fields. Not a name, not a shape, not a dtype. So the bytes are identical in count because the structure never moved. What moved is the values inside the blob.



Herman

Six hundred megs of improvement, no receipts.



Corn

The receipts are there, they're just not itemised on the file. Here's the thing the LoRA paper opens with, and it's basically the founding complaint of this entire subject: the major downside of fine-tuning is that the new model contains as many parameters as the original. The delta you produce has the same dimension as the base model. You do all that training and you walk away with something the same size, which is exactly why LoRA exists as a separate, smaller artifact — people wanted the change saved apart from the model.



Herman

It's worth dwelling on why that complaint existed at all, because it explains the tooling Daniel will encounter. When every fine-tune is a full copy of the model, storage scales linearly with specialisation. Ten people fine-tuning the same base model for ten voices store ten full copies, of which — as we'll get to — almost all the bytes are identical to each other and to the base. The LoRA insight was that if the change is small in rank, you can store just the change: a few skinny matrices instead of a full twin of the model. The merged file Daniel has is what you get when that skinny change gets baked back in, and the baking costs nothing in size because it overwrites in place.



Corn

So when Daniel's file sits there smugly at six hundred and sixty megabytes both times, that's not a mystery. That's the design working exactly as intended. The mystery he's actually feeling is that his ear tells him the model improved, and nothing on the outside of the file confirms it.



Herman

And the answer to that is the task vector. There's a formulation from Ilharco and colleagues, the task arithmetic paper, that I find clarifying. You build the task vector by subtracting the base weights from the fine-tuned weights, tensor by tensor. Delta W. And their definition is lovely: a task vector specifies a direction in weight space such that movement in that direction improves performance on the task.



Corn

A direction. Not a deposit.



Herman

A direction. The improvement isn't an object sitting in the file like a coin in a pocket. It's the direction you'd have to push the base model to get the fine-tuned one. Which is why it's invisible until you subtract — a direction between two points only exists once you draw both points.



Corn

This is the part I want to press on, because I think it's where Daniel's intuition is snagging. He's looking for where the knowledge went in. Like the file should have a new room in it. And your answer is the file is the same house with the furniture turned three degrees.



Herman

Turned, rescaled, and — this is the striking part — barely. The LoRA paper's central hypothesis is that the change in weights during adaptation has low intrinsic rank. Their number: a rank of one or two suffices even when the full dimension is twelve thousand, two hundred and eighty-eight.



Corn

Two directions out of twelve thousand. Can you make that concrete for me? Because rank is one of those words I nod along to and then can't explain at a dinner party.



Herman

Sure. Take a matrix as a machine that transforms whatever passes through it. The rank of the change is, roughly, how many independent things the change does. A rank-one change does one thing everywhere — it's like re-aiming a spotlight. You can move the beam anywhere in the room, make it brighter or dimmer, but it's still one beam. A full-rank change in twelve thousand dimensions would be twelve thousand independent adjustments, each doing something unrelated. The finding is that adaptation looks like the spotlight, not the twelve thousand adjustments.



Corn

Two out of twelve thousand. My own fitness routine has better coverage, and it's theoretical.



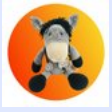
Herman

Yours is theoretical and vertical, mostly. But here's the trap on the other side, because there's a natural instinct to read file size as a learning signal in either direction, and there's a beautiful counter-example in NeMo's own issue tracker. Someone fine-tuned GPT-OSS twenty-billion, and the checkpoint went from about fourteen gigabytes to about forty-two. Tripled.



Corn

Tripling the file for the same model. That's a storage bill that went to the gym.



Herman

And learned nothing from it. The cause was entirely mundane: the weights were MXFP4, the four-bit format, and importing them into NeMo dequantized everything to bfloat16. Nothing new was learned — the numbers just got written out in a fatter encoding. The fix was re-quantizing after the fact. So a file that balloons three times has exactly the same relationship to learning as a file that stays put. Size is not a signal. It's not evidence of learning when it grows, and its stability isn't evidence of failure either.



Corn

It's the same number of facts wearing a coat with more padding.



Herman

Fourteen gigabytes of four-bit numbers and forty-two gigabytes of sixteen-bit numbers can be the very same numbers, the same way "half a kilo" and "one point one pounds" are the same weight on different scales.



Corn

Which cuts both ways for Daniel. His unchanging file is a fact about format, and a reassurance, but the format is also hiding the one number he'd actually want.



Herman

One more piece of housekeeping on that, because it explains a confusion people hit with NeMo specifically. A dot-nemo checkpoint isn't the exported weights. It's a tar bundle — config YAML, the weights, tokenizer artifacts — and the internal checkpoint contains both the model weights and the optimizer states. So a full training checkpoint can be larger than the base model, while the exported weights Daniel is looking at come out the same size. If he ever sees his own file bloat mid-training, that's Adam's extra copies riding along, not knowledge.



Corn

Adam takes luggage. Nobody mentions that in the tutorials.



Herman

They really don't. And the luggage is substantial — the optimizer keeps a running average and a squared-gradient average for every parameter, so during training you're carrying roughly three copies of the model in memory and on disk. It all gets thrown away at export, which is why the final file slims back down and everyone breathes a sigh of relief.



Corn

So, to this point: the file is the same size because nothing about its structure changed, the improvement is a direction in the space between the two files, that direction is small and, in the LoRA finding, shockingly low-rank. Which raises the obvious question — can Daniel actually go and look at it?



Herman

That was my question. He asked whether you can compare the checkpoints and observe what changed and where in the model it changed. Not in principle. On his machine.



Corn

On his machine, CPU only, no model ever runs. And this is the part I got excited about, because there's a small ecosystem of real tools for exactly this. The headline one is called wdiff, weight-diff on PyPI, and its pitch is perfect: git diff, but for neural network weights. You point it at two safetensors files, run it with the deep flag and the lens flag, tell it to write an HTML report, and it streams the tensors on CPU and computes, per tensor, the relative Frobenius norm, cosine similarity, the max change, the average change.



Herman

You've installed it already, haven't you.



Corn

I installed it Tuesday. It took eleven minutes and it worked on the first try, which given my track record with these things —



Herman

Given your track record, the eleven minutes were the setup and the working part is temporary.



Corn

The working part held. And the report is a self-contained HTML file — a topology map of the transformer, a per-layer inspector, everything offline, no CDN, no cloud anything. For a man who spent his working life watching hospital software phone home at inopportune moments, an air-gapped diff report is a small civilisation advance.



Herman

And there are others?



Corn

A few. weightlab, which is early — it's an alpha — but it has a browser-based tensor explorer and a compare dashboard with the same family of metrics. ckptkit, which is the minimal version: it just counts changed tensors versus identical tensors, and honestly that's the one I'd hand someone first if they only wanted a yes-or-no. There's one called wvc that hashes every tensor — gives each one an identity from its bytes — and can diff two Hugging Face repos while downloading only the shards that actually differ. And pentimento, which has the most honest disclaimer in the space. Its change-type classification says, right in the interface, that it's a heuristic, not a calibrated detector.



Herman

I like a tool that confesses.



Corn

It saves you from yourself. But here's the finding buried in wdiff's deep mode that I think is the conceptual heart of this half of the episode. It uses random matrix theory to separate real learning from noise. The idea is that if you took two random matrices of the same shape, they'd have a predictable spectral edge — the Marchenko-Pastur edge — and anything below that edge could just be randomness. So wdiff estimates that edge from the data itself and counts only the singular values meaningfully above it as true signal.



Herman

Meaning if Daniel naively diffs his two checkpoints, tensor by tensor, and looks at how many numbers changed...



Corn

The vast majority of what changed is stochastic gradient descent jitter. Noise. The optimizer shook the whole file on its way past, and nearly all of that shaking carries no learning at all. Only a small number of singular directions sit above the noise floor. So when you said the file is the same house with the furniture turned — the honest version is that nearly all the furniture got jiggled by the truck, and the actual renovation is a few pieces re-aimed.



Herman

That reframes the six hundred megs. It was never six hundred megs of improvement. It was six hundred megs of jiggle with a purpose hiding in it.



Corn

And the lens flag is where it gets almost indecent, because it's the closest thing to actually seeing what the fine-tune did. The static semantic logit lens takes the singular vectors of the weight delta — those few directions above the noise floor — and projects them through the unembedding matrix. Which means it decodes them into plain-English concepts. Which concepts the fine-tune promoted, which it suppressed. Zero forward passes. Zero GPU runtime. The model is never run; the change is read directly.



Herman

So the machine will tell you, in words, what it learned. Quietly, after the fact, without being switched on. I want that for every appliance I own. Herman... hold on. There's a voice I know.

D

Dorothy

Hello dear, it's your mother. Dorothy's History of Toast.

D

Dorothy

In nineteen ninety five a physicist called Robert Matthews published a paper explaining why toast really does tend to land butter side down, mostly because of the height of the table. And the next year it won him an Ig Nobel Prize.

D

Dorothy

So there you are, sweetheart. You spend all that time on the height of a table and the toast still lands butter side down.

D

Dorothy

Anyway. I've got a load of washing on and I should go and see to it. Bye bye now.



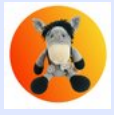
Herman

Mum, you've just described half of machine learning.



Corn

She's on her own schedule, that one. Back to the lens.



Herman

The lens, yes. Though I want to be careful about what it promises, because a promoted concept in the delta is a hint, not a transcript. It'll tell you the fine-tune strengthened the pull toward, say, certain phonetic features — it won't hand you a sentence about Daniel's accent. Think of it like reading the wear pattern on a staircase. You can tell which steps get the most traffic, and roughly which direction people were heading, but you can't reconstruct a single journey from it. What Corn asked about, where in the model the change sits — the report answers that more directly. wdiff also computes something called Grassmannian principal angles between the top subspaces of each layer, and if the biggest angle exceeds twenty-five degrees, it flags that layer as representation drift rather than simple gain scaling.



Corn

Say the second part again for the civilians in the room. One of them is me.



Herman

Two kinds of change live in a weight matrix. The boring kind is gain scaling — everything got a bit louder or a bit quieter, but the same shape. That doesn't change what the layer represents. The interesting kind is rotation — the directions the layer actually uses turned, so the features it computes are different. The principal angles measure that. Below twenty-five degrees, you're mostly turning a dial. Above it, the layer turned a corner.



Corn

So for Daniel the question is which layers turned corners.



Herman

And for a speech model I'd bet — and this is a bet, not a finding — on the lower conformer blocks, the ones closest to the audio, moving most. Speaker identity is an acoustic property before it's a linguistic one. The early layers see the raw waveform features — pitch, timbre, the texture of a particular microphone — before any of it has been squeezed into words. That's where Daniel's voice is most Daniel-shaped. But I'd want the report before I stood behind that.



Corn

You've just handed me the whole toolkit and I notice something. Everything in it measures the change. Nothing in it explains it. We can tell Daniel where his improvement lives, to the tensor, and how much of it is noise, and roughly which concepts it favors. And he still asks the question under the question. How does a fixed array of numbers hold more knowledge just because the values shifted?



Herman

That's the part I've been saving, because the honest answer is strange. The numbers aren't the knowledge. The relationships between them are. A weight matrix is best thought of not as a grid of stored facts but as a set of directions — directions in an activation space the model has carved out during pre-training. Fine-tuning rotates and rescales those directions so that a particular input, Daniel's voice, Daniel's microphone, his pauses, lands in a region of that space the decoder already knows how to read out.



Corn

Nothing new is stored.



Herman

The mapping is re-aimed. The decoder already had the vocabulary. Pre-training gave Parakeet a hundred and twenty thousand hours of English speech — ten thousand human-transcribed, a hundred and ten thousand pseudo-labelled — and then NVIDIA's own final polish was twenty-five hundred steps on four A100s with about five hundred hours. Twenty-five hundred steps. That stage-two fine-tune is a rounding error against the pre-training, and it still moved the leaderboard numbers. That ratio is the whole intuition: the capacity was already there, bought with a hundred and twenty thousand hours, and fine-tuning spends a sliver of a percent of that budget re-pointing it.



Corn

And I want to underline that ratio, because it's the strongest evidence we have that nothing new gets stored. If fine-tuning were about acquiring knowledge, the amount learned should scale roughly with the amount of data. It doesn't. NVIDIA's own polish run took a model that had heard a hundred and twenty thousand hours and improved it with five hundred — less than half of one percent more data — and it moved the numbers. You cannot learn much from half a percent more of anything. You can only re-aim what's already there.



Herman

You said something earlier I want to pull back, because I think it's the actual answer to his conceptual question. You said the base model already knows how to read the region. Why does it have spare regions lying around?



Corn

There's a paper I keep thinking about, from this year's ICML — the short version is expand neurons, not parameters. The finding is that you can improve a network's accuracy by increasing its neurons without increasing its non-zero parameters. Which sounds like a trick, until you see why: the gain corresponds to a decrease in interference between features that were sharing the same neurons. And the line that rewired my thinking is that even random splits of the neuron weights approximate these gains. Reduced collisions, not precise assignment, are the primary driver.



Herman

Hold on. Random works?



Corn

Nearly as well as clever. Which tells you the win isn't from finding the perfect place for each feature — it's from features not being crushed together. Polysemanticity, they call it — one neuron carrying several unrelated features, blurring them into each other. Accuracy gains are largest exactly where that crowding is worst.



Herman

So then Daniel's fine-tune isn't adding knowledge of his voice. It's giving his voice room.



Corn

That's how I'd phrase it now, yes. His voice was being crushed together with everyone else's in shared neurons — the model heard him through the crowd. A small personal dataset, well-aimed, relieves the interference. The information about his speech was partly present and partly colliding, and the fine-tune de-crowds it. That's why a few hours of his audio can matter at all. Nobody learns speech from a few hours. But a few hours is plenty to shove one speaker out of the crush.



Herman

That's the answer I'd give him over dinner, I think. The file doesn't hold more. It holds the same numbers, arranged so his voice stops being misread as an average of other people.



Corn

There's a companion result on why a small nudge is even possible, from a paper on intrinsic dimension. The toy example: a thousand-dimensional parameter space where the set of good solutions is nine hundred and ninety dimensions wide. From any point that already works, there are nine hundred and ninety orthogonal directions you can move and stay working. The intrinsic dimension — the number of things you actually need to get right — is ten.



Herman

Ten.



Corn

Ten, in a thousand. The space is almost entirely solution. And their general conclusion: once a parameter space is large enough to solve a problem, extra parameters serve to increase the dimensionality of the solution manifold — they make the good region bigger and easier to hit from anywhere. Six hundred million parameters is not six hundred million facts. It's an enormous, almost-all-solution space with pre-training holding you inside it.



Herman

Which is why his nudge doesn't break the model. You can't wreck the dinner when most of the kitchen is edible.



Corn

I'd frame it as: you'd have to try to miss. And there's a party trick that proves the delta is a real, manipulable object rather than a metaphor — task arithmetic. Ilharco's group showed you can negate a task vector, subtract it from the fine-tuned model, and you get the task un-learned, with little change to everything else. Add two task vectors, you get both capabilities. And in the analogy case — A is to B as C is to D — combining three task vectors improves the fourth even with no training data from the fourth task at all.



Herman

So if Daniel's voice adaptation is a proper task vector, subtracting it hands him back the generic model.



Corn

Testable, on his hardware, today. That's the demo I want him to run. The reason it works at all is a property called weight disentanglement — distinct directions in weight space govern separate regions in function space — and the follow-up work shows that property is a general consequence of pre-training. The hundred and twenty thousand hours didn't just teach the model English. It organised the weight space into separable directions, so a push along the voice direction doesn't smear the grammar direction.



Herman

It's a well-sorted toolbox before anyone arrives with the new screwdriver.



Hilbert

Herman, your Grassmannian angles — you explained those better than the paper does, I'll give you that. But I've run enough of these comparisons to know a count tells you nothing.



Herman

A count?



Hilbert

A customer of mine, Mrs. Aldous, up on Cherrywood Road, wanted two versions of her voiceover kept straight — the original and a re-record she'd paid for. She was certain the re-record had fixed a lisp she swore was in the first take. I compared them for her. Byte counts matched to the megabyte, waveform lengths matched, so I told her nothing had changed. She took that well, meaning she didn't.



Corn

Meaning it had changed.



Hilbert

I'd compared the re-record against a second copy of the re-record. Both files had her handwriting on the label, her handwriting being what it is. The lisp was fixed in the version I never measured. Mrs. Aldous has the pair of them now, in the folder with her name spelled wrong on the tab, which I've left as it is because she knows which one is hers.



Corn

So the diff was perfect. The inputs were wrong.



Hilbert

The diff was perfect. That's my whole point, Corn. Any of these tools you were praising — the ones that say four hundred tensors changed, six didn't — that's the version of the story that sends Mrs. Aldous home certain her lisp is still there. The count is the least interesting number the two files produce. Your noise-floor tool is the only one of the lot saying anything worth the electricity, because it separates the two directions that matter from the truck jiggle. And I'll say — what you said about the model knowing how to read the region already. That one's been sitting with me since you said it. I'd been treating her two takes as different objects. They weren't. Same voice, re-aimed.



Herman

So you'd tell Daniel to check his two files are actually his two files before he reads the report.



Hilbert

I'd tell him the labels are half the comparison, and that's me being generous to the labels. Anyway. Mrs. Aldous rang Tuesday about a third take. There's no lisp in that one either, but she's found something else.



Corn

The photograph metaphor just gained a second photograph taken by mistake.



Herman

It did, and it's a fair correction to the enthusiasm. I'd still run the diff — but Corn, you were the one who said the tools measure the change without explaining it. Hilbert's version is harsher: they measure whatever you point them at, and they don't check you pointed well.



Corn

Which is why the two questions Daniel asked are really one question asked twice. Where did the improvement go, and can I see it — both of them resolve the same way. Subtract, carefully, and most of what you find is noise.



Herman

The honest open questions are the ones worth leaving him with. Nobody's measured the intrinsic dimension of ASR fine-tuning or speaker adaptation specifically — the intrinsic-dimension work covers images and reinforcement learning, not speech. So how low-rank is his voice adaptation, really? His own report would tell him. The noise-floor tool would put a number on it, and I'd love to know whether it's closer to the LoRA one-or-two end or the dozens.



Corn

The config. Whether his recipe was a full fine-tune or a merged adapter, because that decides whether the low-rank delta is a design guarantee or an empirical surprise.



Herman

He checks the YAML, he knows which of two stories he's holding. One last thing before we go, because it's the image I want Daniel left with. The file is a photograph. The improvement is the difference between two photographs — it has no existence of its own until you subtract, and most of what the subtraction finds is jiggle. If you're enjoying this sort of thing, a review wherever you get the show helps more than you'd think. Producer credit to Hilbert Flumingtop, who keeps the desk, the labels, and apparently Mrs. Aldous, all in working order. If this was your kind of episode, go back for episode eleven, How Does Fine Tuning Work Anyway; episode three, Safetensors or something else; and episode four, If Your Voice Ages, Does Your Fine-Tune Become Useless. This has been My Weird Prompts — find us at my weird prompts dot com. See you tomorrow, brother.



Corn

See you tomorrow. And Daniel — keep both files. Label them yourself.

SOURCES

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

1. **NVIDIA, Hugging Face model card, created 2025-04-15, last modified 2026-06-29, retrieved 2026-11-19. [primary]**
   https://huggingface.co/nvidia/parakeet-tdt-0.6b-v2

2. **NVIDIA NeMo Framework User Guide, Checkpoints, retrieved 2026-11-19. [primary]**
   https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/checkpoints/intro.html

3. **NeMo issue #14619, Huge size bump for GPT-OSS-20b after fine tuning from 14GB to 42GB, created 2025-09-02, updated 2026-02-26, retrieved 2026-11-19. [primary]**
   https://github.com/NVIDIA-NeMo/NeMo/issues/14619

4. **NVIDIA NeMo AutoModel, SFT and PEFT, retrieved 2026-11-19. [primary]**
https://docs.nvidia.com/nemo/automodel/latest/recipes-e2e-examples/sft-peft

5. **`wdiff` / weight-diff v0.3.0, PyPI, retrieved 2026-11-19. [primary]**
https://pypi.org/project/weight-diff/

6. **weightlab v0.1.0a2, PyPI, retrieved 2026-11-19. [primary]**
https://pypi.org/project/weightlab/

7. **ckptkit, retrieved 2026-11-19. [primary]**
https://github.com/stef41/ckptkit

8. **wvc, retrieved 2026-11-19. [primary]**
https://github.com/evnsnclr/wvc

9. **pentimento, retrieved 2026-11-19. [primary]**
https://github.com/chayan-bit/pentimento

10. **Ilharco, Ribeiro, Wortsman et al., Editing Models with Task Arithmetic, ICLR 2023, v3 revised 2023-03-31, retrieved 2026-11-19.**
https://arxiv.org/abs/2212.04089v3

11. **Hu et al., LoRA: Low-Rank Adaptation of Large Language Models, retrieved 2026-11-19.**
https://arxiv.org/abs/2106.09685

12. **Li, Fusi, Hochreiter et al., Measuring the Intrinsic Dimension of Objective Landscapes, retrieved 2026-11-19.**
https://arxiv.org/abs/1804.08838

13. **Ortiz-Jimenez, Favero, Frossard, Task Arithmetic in the Tangent Space, NeurIPS 2023, v3 revised 2023-11-21, retrieved 2026-11-19.**
https://arxiv.org/abs/2305.12827v3

14. **Hazimeh, Favero, Frossard, Task Addition and Weight Disentanglement in Closed-Vocabulary Models, submitted 2025-11-18, retrieved 2026-11-19.**
https://arxiv.org/abs/2511.14569v1

15. **Kong, Subramanian, Shavit et al., Expand Neurons, Not Parameters, ICML 2026, v3 revised 2026-06-03, retrieved 2026-11-19.**
https://arxiv.org/abs/2510.04500v3

16. **Elhage, Hume, Olsson et al., Toy Models of Superposition, 2022-09-21, retrieved 2026-11-19.**
https://arxiv.org/abs/2209.10652v1

17. **Ait Hou & Hwa, Polysemanticity or Polysemy? Lexical Identity Confounds Superposition Metrics, 2026-04-01, retrieved 2026-11-19.**
https://arxiv.org/abs/2604.00443v1

18. **Sasaki & Sato, On the Sparsity of Direct Preference Optimization, 2026-10-27, retrieved 2026-11-19.**
https://arxiv.org/abs/2610.37085v1

19. **Iurada, Ciccone, Tommasi, Efficient Model Editing with Task-Localized Sparse Fine-tuning, ICLR 2025, retrieved 2026-11-19.**
https://arxiv.org/abs/2504.02620v1

20. **Wikipedia, last edited 2026-11-09, retrieved 2026-11-19 (background on tensor-container formats).**
https://en.wikipedia.org/wiki/GGUF