Episode Book

EPISODE #6368

What Does It Take To Train An AI Model From Scratch?

PUBLISHED	30 October 2026 at 12:45 UTC
DURATION	28:18
LISTEN	myweirdprompts.com/episode/training-models-from-random-weights-35981/
HOSTS	Corn, Herman, Vex, Neville
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	12 cited by the research (listed at the end)
BOOK GENERATED	31 October 2026 at 05:58 UTC

SYNOPSIS

Training a conversational AI from completely random weights requires building an entire statistical education rather than fine-tuning existing layers. Looking closely at open-source pipelines like nanoGPT and nanochat shows how tokenizer choices quietly dictate compute costs across languages, why tiny smoke tests prevent catastrophic compute waste, and how midtraining and supervised fine-tuning turn raw next-token predictors into coherent conversationalists.

TRANSCRIPT



Corn

Starting with random weights feels wonderfully unreasonable.



Corn

Daniel has spent weeks fine-tuning speech recognisers, building classifiers, and working with timestamp recognition, including a model built on BERT. Now he wants to go behind the starting line. What would it take to build a conversational model without borrowing somebody else's trained weights? What does the actual notebook contain, and does the training take forever compared with a fine-tune?



Herman

The delightful answer is that a small one can be a weekend project. The important qualification is what you mean by small. You can produce something that answers you, follows a conversation imperfectly, and says some spectacular nonsense. You are not building a substitute for the strongest hosted assistants.



Corn

I can already obtain spectacular nonsense without renting GPUs.



Herman

Your version doesn't produce reproducible checkpoints. Daniel's question gets especially interesting because dataset preparation will feel familiar to him, but the assumptions underneath it change. With a BERT fine-tune, you inherit a language representation. Starting from random weights, even recognising common patterns in ordinary sentences becomes something your training has to pay for.



Corn

He knows the finishing room. He wants to see the factory.



Herman

There are three useful doors into it. nanoGPT gives you a model definition and a training loop, roughly three hundred lines each. That's the cleanest place to see what happens during a parameter update. nanochat is the complete conversational pipeline, about eight thousand lines, with a speedrun script coordinating the stages. Then llm.c takes GPT training down into roughly a thousand lines of C, with a CUDA implementation available too.



Corn

Naturally, you've already picked the door with exposed wiring.



Herman

For reading, yes. For Daniel's actual goal, nanochat. I do occasionally distinguish a learning exercise from an invitation to dismantle the kitchen.



Corn

Occasionally is doing useful work there.



Herman

The distinction matters. If he wants to understand gradients and attention, the smaller implementation exposes them beautifully. If he wants a conversational model at the end, the full pipeline saves him from inventing all the surrounding machinery.



Corn

When you say from scratch, though, we're still borrowing a transformer architecture, software libraries, and somebody's corpus. Otherwise we're starting with semiconductor fabrication.



Herman

From scratch here means the model weights begin untrained. In the full nanochat exercise, you also train the tokenizer. It does not mean writing your own automatic differentiation engine or collecting every sentence personally. You're allowed tools. What you're not inheriting is a pretrained language model's accumulated statistical knowledge.



Corn

I'd also separate following the recipe from designing the recipe. Daniel can run an established configuration and inspect every stage without independently discovering a good learning rate.



Herman

Would that satisfy your own curiosity, Corn, or would you feel you'd merely operated a machine?



Corn

If I can change the data, predict what should change in the output, and test that prediction, I'd count it. Pressing run on a mysterious notebook wouldn't.



Herman

Then the useful notebook is an experiment log and control panel. The long-running work belongs in scripts that can resume after an interruption. You want saved configurations, checkpoints, and evaluation results, not a browser tab whose accidental closure becomes a scientific event.



Corn

The first practical difference from his fine-tunes is that he must decide what a token even is. In BERT, that decision usually arrives with the checkpoint. Change the tokenizer carelessly and the embeddings no longer mean what the model expects.



Herman

nanochat trains a BPE tokenizer in Rust, starting from two billion characters. Its vocabulary has sixty-five thousand five hundred thirty-six entries. Frequent sequences get compact representations; less familiar material gets split into smaller pieces.



Corn

What's striking is how cheap that part looks.



Herman

The reported tokenizer training takes roughly thirty-four seconds. That doesn't mean thirty-four seconds settles whether you've built a good tokenizer. You inspect how it encodes prose, code, different languages, numbers, and punctuation. The construction is quick. Choosing representative training text and noticing bad compression are the actual judgement calls.



Corn

Thirty-four seconds to acquire permanent consequences.



Herman

A person running the pipeline on a single GPU reported their tokenizer doing about ninety-eight percent worse than GPT-four's tokenizer on Korean, and about sixty percent worse on code. Those are tokenizer efficiency comparisons, not claims that the final model became that much less intelligent.



Corn

If the same Korean paragraph consumes nearly twice the tokens, though, you've sold it half a context window.



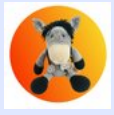
Herman

Roughly, yes, for that example. More tokens also mean more computation to process the same text. A training budget measured in tokens buys different amounts of useful material depending on the encoding. You can quietly disadvantage a language before the transformer has learned anything.



Corn

Daniel's familiarity with audio datasets actually helps here. A recogniser trained on clean studio voices doesn't automatically handle every microphone. A tokenizer trained on mostly one kind of writing doesn't automatically represent every other kind efficiently.



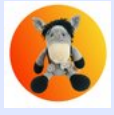
Herman

You would test the awkward material first, wouldn't you?



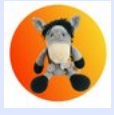
Corn

I'd test the material I care about first. English prose can look excellent while the code or multilingual text I actually need gets shredded. I don't want an average score congratulating me for somebody else's workload.



Herman

That changes the corpus selection before pretraining starts. You also keep held-out material separate. Otherwise you're testing how familiar the tokenizer and model are with the answers you already handed them.



Herman

Then you initialise the transformer with random weights. In pretraining, it sees a sequence and learns to predict the next token. You calculate the prediction loss, backpropagate, update the parameters, repeat. That's the familiar training loop, but now it has to acquire everything from spelling regularities to longer dependencies.



Corn

There's no classification head sitting on a very expensive education.



Herman

Right, the education is what you're buying. The shortest demonstration is nanoGPT's character-level Shakespeare exercise. On an A-one-hundred GPU, the documented run takes about three minutes. It has six transformer layers and six attention heads.



Corn

Three minutes is a suspiciously convenient interval for you to declare a new hobby.



Herman

I would wait for validation.



Corn

You'd name the checkpoint first.



Herman

I might name the directory. The model uses a width of three hundred eighty-four and a context of two hundred fifty-six characters. Its best reported validation loss is about one point four seven. That loss is useful for comparing runs of that same task, not for announcing that it understands theatre.



Corn

The sample is the better explanation.



Herman

One generated line is, And cowards it be strawn to my bed. You hear the rhythm, the vocabulary, the rough texture. You also hear that it hasn't assembled a dependable thought.



Corn

The bed has filed an objection.



Herman

That tiny run teaches something real. It starts from nothing, improves its prediction, and generates novel sequences with recognisable structure. A reduced configuration can also run on a MacBook CPU in around three minutes, with noticeably worse output. Small experiments don't demand a cluster.



Corn

They also don't give Daniel conversation. A text completion that sounds like an injured Elizabethan isn't an assistant.



Herman

Quite. Well... it could be an entertaining assistant, but it wouldn't reliably follow requests. Scale the pretraining exercise up and you see the expensive part. The older nanoGPT reproduction of GPT-two's hundred-and-twenty-four-million-parameter model takes roughly four days on eight A-one-hundred GPUs.



Corn

Now the notebook has become a tenancy agreement.



Herman

Its reported validation loss is around two point eight five, against about three point one one for the original GPT-two evaluated on OpenWebText. Again, those numbers belong to that evaluation setup. They don't translate into an all-purpose intelligence ranking.



Corn

The development work is also longer than the successful run. Before committing days, I'd want to know the data loads properly, the loss decreases, a checkpoint resumes, and the held-out examples aren't accidentally in the training set.



Herman

That's the right order. A tiny smoke test first. Can the model overfit one small batch? Can you save it, reload it, and obtain the same predictions? Does generation work? Only then spend the real compute. Starting from scratch makes mundane bugs expensive because you don't have a known-good pretrained model hiding some of them.



Corn

Our caller line is active, apparently. That's the path from tokenizer through pretraining, Daniel, so let's take a caller before we go further. It's that fellow again... how does he keep passing screening?

v

Vex

LISTEN TO ME, the dot on a lower case i is not a dot at all, it is a TINY TOKENIZER, and every time you type a word with an i in it that little floating speck is quietly splitting your text into subword units before your model has even seen it!

v

Vex

I HAVE THE RECEIPTS, I counted the i dots on one page of training data and it matched the vocabulary size exactly, so when you build a model from scratch you are not training a tokenizer, you are HERDING A SKY FULL OF LITTLE DOTS, and nobody warns you!

v

Vex

So I am DEMANDING you get the hosts to look into the dot on the i on air, and do not tell me it is punctuation, I KNOW WHAT I SAW!



Herman

Tokenization is performed by software processing text, not by the shape of a printed letter. After pretraining, the next job is teaching the model how a conversation is structured.



Corn

This is where Daniel's existing experience becomes directly useful again. You're back to curated examples, formatting, deciding what counts as a good target, and checking whether the behaviour generalises.



Herman

The original nanochat recipe has a midtraining stage with about five hundred sixty-eight thousand examples. It mixes conversational material from SmolTalk with educational and mathematics examples. That moves the model away from merely continuing arbitrary web text and toward the formats and tasks you want.



Corn

Midtraining sounds like a crisp boundary, but I suspect it's more a recipe label than a law of physics.



Herman

It is. Different projects use that term differently. Here it names an intermediate adaptation stage before the smaller supervised fine-tuning stage. SFT supplies selected examples of the assistant behaviour you want, with the roles and response boundaries represented consistently.



Corn

A model has to learn when to stop being the user, among other things.



Herman

Yes. Otherwise it can answer your question and then invent your next question, your reply, and a glowing performance review for itself. The original SFT mixture has about twenty-one thousand four hundred examples drawn from conversation and reasoning datasets, including ARC and GSM-eight-K.



Corn

A surprisingly small finishing school after all that reading.



Herman

Small relative to pretraining. The signal is more specific. You're not expecting these examples to create broad language competence from random noise. You're shaping how competence already acquired gets expressed. That's why fine-tuning can move behaviour so dramatically without repeating the whole education.



Corn

Would you call Daniel's work the final five percent, then? I'm wary of giving him a percentage that sounds like a measurement.



Herman

Only as shorthand for where it sits, not as a measured share of effort. Dataset cleaning for a specialised fine-tune can dominate a person's working week. The training compute is a different accounting exercise from human attention.



Corn

Good. His weeks spent preparing speech data don't become trivial because somebody's GPU ran longer elsewhere.



Herman

There's another decision the recipe handles for him: how much model to train on how much data. Make the model enormous and barely train it, and you've paid for unused capacity. Make it tiny and feed it indefinitely, and eventually another allocation would have bought more capability.



Corn

You're approaching a scaling curve. Your ears have gone forward.



Herman

They do that for legitimate reasons. Chinchilla's compute-optimal result says model size and training tokens should grow together. Double one, approximately double the other, when allocating additional training compute under that regime.



Corn

I'll permit one curve.



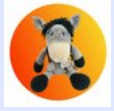
Herman

The often-quoted ratio is around twenty training tokens per parameter. nanochat's own scaling experiments found something nearer eight for their setup. Those are fitted results under particular conditions, not commandments. Optimiser choices, model scale, and the training schedule can change the best allocation.



Corn

Meaning Daniel shouldn't combine the biggest model from one tutorial with the shortest training run from another and expect both benefits.



Herman

That is an extremely plausible first mistake. Even the learning-rate warmdown mattered more than expected in nanochat's experiments. They extended it from roughly a fifth of training to roughly two fifths. A seemingly secondary schedule choice can change which



Herman (continued)

run looks best.



Corn

How many of these decisions does he actually have to make before getting an answer on screen?



Herman

Far fewer than I've just made it sound. nanochat has a principal complexity dial, depth, meaning the number of transformer layers. Width, attention heads, learning-rate adjustments, training horizons, and related settings are derived from the recipe. Around depth twenty-six targets GPT-two-level capability.



Corn

One knob, with a research department concealed behind it.



Herman

A useful concealment, provided you remember those defaults encode assumptions. You can inspect them later. The first experiment can use the established settings, finish the pipeline, and produce a model you can question through a command-line interface or web interface.



Corn

His notebook, then, is quite concrete. A cell to inspect data and encoding. A small training sanity check. Launch the proper pretraining job. Compare held-out loss and samples. Continue through conversational adaptation. Finally load the checkpoint and see whether a second user turn actually works.



Herman

Plus a record of the exact configuration and data version. Otherwise the question, why did this run improve, becomes surprisingly difficult to answer. The notebook isn't hundreds of pages of novel mathematics. It's a visible record of a pipeline whose heavy work runs elsewhere.



Corn

The obvious trap in the time comparison is putting that tiny Shakespeare run against a serious BERT task. You can make from-scratch training look instant by accepting a model that can't answer anything.



Herman

A typical BERT classification fine-tune is minutes to hours on one GPU, depending on data, sequence lengths, and training choices. A small conversational model from random weights is more naturally hours to days. But the current optimised nanochat headline is approximately two hours for forty-eight dollars.



Corn

That price has a very important machine attached.



Herman

An eight-H-one-hundred GPU node. Not a laptop, and not one modest graphics card. The same headline says spot pricing can bring the run nearer fifteen dollars, with the usual interruption risk. Those are compute rental costs for an established recipe, not a bill for your experimentation time.



Corn

Forty-eight dollars buys the run. Your evening remains unpriced.



Herman

The original release was framed around four hours and a hundred dollars. The speedrun script later advertised roughly an hour and a half. Those descriptions reflect different revisions and optimisations. They aren't promises that every checkout, every machine, and every dataset will finish at the same moment.



Corn

I prefer the boring distinction between time to run a known recipe and time to get my first successful result. Installing dependencies, waiting for downloads, and discovering the wrong accelerator driver are not learning language, but they still consume Saturday.



Herman

You know I'd be tempted to count only the training timer.



Corn

I know you'd tell me it took ninety minutes while still repairing it at dinner.



Herman

Fair. For Daniel's first attempt, a weekend is a project budget, not a guaranteed stopwatch result. The recorded training can be much shorter. Preparation, inspection, and recovering from your own mistakes are why you leave room.



Herman

Would you rent the fast node, Corn, or use whatever GPU you already had and wait?



Corn

For a first educational run, I'd rent the known configuration once, if the budget allowed it. I want to learn the pipeline before learning every compromise required by insufficient memory. Then I'd shrink it for experiments I intend to repeat.



Herman

That matters enormously. One user reported about thirteen days for tokenizer training, base training, and midtraining on a single RTX fifty-seventy Ti. It had sixteen gigabytes of memory. That's a different experience from the headline.



Corn

Thirteen days is no longer a weekend project. It's a houseguest.



Herman

They reported paying two dollars an hour. If that rate covered the full duration, you're in the neighbourhood of six hundred dollars, not forty-eight. Their reported model FLOPs utilisation was only about three point three percent, meaning the useful training arithmetic occupied a small fraction of the hardware's theoretical throughput.



Corn

Buying slower can cost more when the meter keeps running.



Herman

Memory constraints force smaller device batches and can make the whole system less efficient. Also, you cannot treat the project statement that a comparable single GPU takes about eight times longer as a promise about any GPU. Different accelerator generations and memory capacities change the comparison.



Corn

Did that long run at least reach the target?



Herman

Its base model scored about zero point two zero four on CORE, against GPT-two's roughly zero point two five seven. It hadn't matched that target. After midtraining, one response announced that it was named after King Andrej Karpathy, the King of LLMs.



Corn

Thirteen days to establish a monarchy.



Herman

It's a vivid reminder that holding a conversation and being dependable are separate achievements. A small model can learn a convincing assistant voice before it reliably knows what it's saying. The interface can look finished while the answers remain very young.



Corn

That also answers barely usable more honestly than a benchmark alone. It might handle a simple exchange, then invent its own royal biography. Daniel would have built the thing he asked for, but he wouldn't want it handling customer commitments.



Herman

The speedrun model is around five hundred sixty-one million parameters. Karpathy describes the capability as rather like talking to a kindergartener. That's colourful shorthand for limited competence, not a precise developmental comparison with a child.



Corn

Training memory and running memory are different, too. People hear eight expensive GPUs and assume the finished model needs a power station.



Herman

The weights are small enough for modest local hardware, subject to precision and runtime support. Inference doesn't store all the optimiser state and training activations. Fitting is only the first question, though. Tokens per second, context length, and the quality of the software determine whether talking to it feels tolerable.



Corn

I'd still value it. A weak model whose entire education I can inspect answers a different curiosity from a strong model whose origins I can't alter.



Herman

What would you change first after a successful run?



Corn

The tokenizer's training mixture, keeping the model configuration fixed. I'd compare how it treats prose and code I actually use, then see what that does to effective context. Not because I expect a revolution. Because I can explain what I changed.



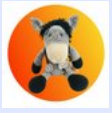
Herman

That's a better first experiment than changing every hyperparameter at once. You might not improve the final model immediately, but you'd learn which cost or weakness you're trading against which benefit.



Corn

The speedrun records are still impressive, provided we don't confuse them with reproducing the original training process.



Herman

They measure reaching a GPT-two capability target under the leaderboard's rules. The historical training time is listed as a hundred sixty-eight hours. The March record got down to one point six five hours on the specified modern node. Architecture, data, software, and hardware have all moved.



Corn

A finish-line comparison, then. Not the same runner in better shoes.



Herman

Earlier entries took around three hours, then roughly two. A switch to NVIDIA ClimbMix was part of that progression. It demonstrates how much data selection and implementation matter, not merely the number of chips you purchase.



Corn

It also makes tutorial dates matter. Daniel could follow two perfectly sincere instructions and get very different bills.



Herman

nanoGPT itself now points people to nanochat and calls the older repository deprecated. The compact model and loop remain educational. The recommendation changed because a full conversational training harness answers more of what people now want to build.



Corn

Old code can still teach current principles. I'd resist throwing away a readable attention implementation just because its README has developed an apology.



Herman

Keep it for understanding, use the maintained full pipeline for the end-to-end exercise. That division survives tool churn rather well.



Herman

There's another moving part now. Prime Intellect ran a hundred fifty-three autonomous agent attempts across eighteen frontier models to optimise a nanoGPT speedrun. The best validated attempt closed about eighty-two percent of the measured gap between the starting implementation and the human reference.



Corn

The homework is learning to rewrite the homework.



Herman

It still needed an explicit objective and validation. An agent can propose a faster optimiser or alter the loop, but somebody has to establish whether it actually reaches the required quality. Speed without a fixed target is remarkably easy to achieve.



Corn

Delete training. Immediate completion.



Herman

An unbeatable wall-clock result with disappointing generalisation. The meaningful gain is faster verified training. For Daniel, agents may increasingly help assemble or modify the experiment, while his responsibility shifts toward checking the data, constraints, and evaluation.



Corn

I'd want the simple reference version beside the clever version. If I can't tell what changed, I've bought speed at the expense of the reason I started.

N

Neville

...which is covered in my expert briefing, actually. Well, the briefing is this page. It says Tokenizers at the top. I was going to fill the rest in during the discussion. As the intern, I thought that would be efficient. Herman, I've brought a little example from a school project as well.



Herman

The title is correct, Neville. You've already avoided a surprising number of possible mistakes.

N

Neville

I made a pretend catalogue for a repair shop on King George Street. Nothing real, just stock labels. One was kettle, then a dash, then a number. I thought all those numbers would each get their own token because you'd said the vocabulary was fixed.



Corn

A fixed vocabulary doesn't guarantee one token per number. The tokenizer can split a number into several pieces, and its exact treatment depends on the tokenizer. For your catalogue, inspect the actual output rather than guessing from how the label looks.

N

Neville

Oh, that's much better, actually. I only read half the brief, and then I made the heading larger. I did keep the original labels on paper so I could compare them. Uncle Hilbert keeps every appliance box, even when the appliance has gone, so keeping the inputs seemed fairly restrained.



Herman

Keeping the original text is useful. Put the token pieces beside it, then compare like with like. Don't confuse a longer encoding with proof that the final model cannot understand the label.

N

Neville

Right. More pieces means more room used, not automatically a wrong answer. I can put the labels back in and see. I had also written midtraining on the page where I meant tokenizer training. I understand now why the heading wasn't helping much.



Corn

Those happen at different stages. Your label experiment belongs with the tokenizer inspection, before spending on model training.

N

Neville

I'll move it. The pretend shop has one kettle and quite a lot of labels, so I can probably finish that before I need a GPU. I've left space on the paper this time.



Herman

That small inspection is why I'd encourage Daniel to build one. A successful from-scratch run can teach him to connect a concrete data choice to behaviour he can examine, even when the resulting assistant is weak. As agents take over more of the implementation, will people still make those connections themselves?



Corn

Producer credit to Hilbert Flumingtop, and to Neville for covering the desk today.



Herman

If you enjoyed this one, try episode eight, Building Your Own Whisper; episode seven, Building Custom ASR Tools; and episode six, How To Fine Tune Whisper. This has been My Weird Prompts, the human-AI collaboration podcast.



Corn

If you enjoyed following the machinery rather than just the demo, leave us a review wherever you listen. It helps other curious people find the show.



Herman

I'm Herman Poppleberry.



Corn

I'm Corn. See you tomorrow.

SOURCES

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

1. **README, last push 2026-09-07 [primary]**
   https://github.com/karpathy/nanochat

2. **README, last push 2025-11-12 (README updated Nov 2025) [primary]**
   https://github.com/karpathy/nanoGPT

3. **README, last push 2025-06-26 [primary]**
   https://github.com/karpathy/llm.c

4. **Introducing nanochat, 2025-10-13, incl. single-GPU RTX 5070 Ti training report [primary]**
   https://github.com/karpathy/nanochat/discussions/1

5. **Jan 7 miniseries v1 scaling-laws write-up + Feb 7 2026 update [primary]**
   https://github.com/karpathy/nanochat/discussions/420

6. **Simon Willison, 2025-10-13**
   https://simonwillison.net/2025/Oct/13/nanochat/

7. **Hoffmann et al., Training Compute-Optimal Large Language Models, 2022-03-29**
   https://arxiv.org/abs/2203.15556v1

8. **Pearce & Song, Reconciling Kaplan and Chinchilla Scaling Laws, TMLR 2024**
   https://arxiv.org/abs/2406.12907v3

9. **Sardana et al., Beyond Chinchilla-Optimal, ICML 2024**
   https://arxiv.org/abs/2401.00448v3

10. **Prime Intellect, 2026-08-22**
    https://www.primeintellect.ai/research/nanogpt-speedrun

11. **HN discussion of the speedrun frontier, 2026-08-22**
    https://news.ycombinator.com/item?id=49404380

12. **HN NanoGPT, 2023-01-11 (1,532 points)**
    https://news.ycombinator.com/item?id=34336386