Episode Book

EPISODE #6723

Why Weight Initialisation Depends on Architecture, Not Task

PUBLISHED	21 November 2026 at 13:30 UTC
DURATION	33:13
LISTEN	myweirdprompts.com/episode/neural-network-weight-initialisation-schemes-52077/
HOSTS	Corn, Herman, Hilbert, Larry
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	21 November 2026 at 19:31 UTC

SYNOPSIS

Random weight initialization is not a one-size-fits-all roll of the dice dictated by the machine learning task. The math behind starting weights depends entirely on activation functions, normalization, and network depth. From classic LeCun and He formulas to orthogonal matrices, data-dependent scaling, and deterministic biases, exploring how models like nanoGPT balance signal variance shows why architecture dictates the numbers before training even begins.

TRANSCRIPT



Corn

Every from-scratch story starts the same way. Some paper, some repo, some blog post says step one: initialise the weights randomly. Two words doing about four decades of unpaid overtime. Daniel's been on Hugging Face lately, and he noticed the model families there don't all wear the same kind of randomness, so his question to us is: if you're training a language model versus an ASR model, do you actually start from the same kind of random weights, or does each kind of task carry its own approach to initialisation?



Herman

And it's a good thing he asked, because the short answer is no, and the interesting answer is that the reason has almost nothing to do with the task. The dividing line is architecture, activation function, normalisation, depth. Task is the label on the box; initialisation is decided by what's inside the box.



Corn

He's going to say "it's not about the task" about six more times before we're done.



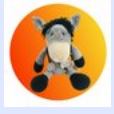
Herman

I am, because it's true, and because once you see it you can't unsee it. The premise is that somewhere there's an ASR initialisation and an LLM initialisation, sitting in separate drawers. In reality there's a spectrum, and half of it isn't even random.



Corn

Let's set the table, then. What does "start from randomised weights" actually mean, mechanically?



Herman

Before the first gradient step, every trainable parameter in the network needs a value. Weights and biases, all of them. "Initialisation" is just the act of assigning those starting values. You can't start at zero, and that's the part people find counterintuitive. If every weight is zero, every neuron in a layer receives the same input, produces the same output, gets the same gradient, and updates identically. Forever. They're perfectly symmetric, so they all learn the same feature and you've paid for a wide network and gotten a narrow



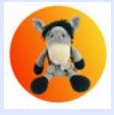
Herman (continued)

one.



Corn

A hundred workers who all read the same memo.



Herman

And here's the perverse part — it's not just that you get one feature instead of a hundred. It's that no amount of training fixes it, because the symmetry is self-perpetuating. Identical inputs, identical gradients, identical updates, at every step, for the entire run. The only thing that could break the tie is the randomness you refused to provide. People sometimes ask whether you could break the symmetry some other way — randomise the data order, say — but the data arrives through the weights, so if the weights are identical, each neuron sees the same data the same way. The tie has to be broken inside the layer itself.



Corn

So randomness is load-bearing, not decorative.



Herman

Randomness is there to break the symmetry, that's its first job. Its second job is scale. Your initial values determine the magnitude of activations flowing forward and gradients flowing backward, and if either of those shrinks or blows up layer by layer, by layer twenty there's nothing left to learn from. Vanishing and exploding gradients, activation saturation. Initialisation is the first line of defence against all of it, before normalisation even gets a chance.



Corn

Give me the failure in slow motion. What does "shrinks layer by layer" actually look like?



Herman

Picture each layer multiplying the signal by some average factor. If that factor is 0.9, after a hundred layers you're at 0.9 to the hundredth — you've multiplied your signal by roughly three hundredths of a percent of what you started with. The gradient that reaches the early layers is a rounding error, so those layers learn at a crawl while the late layers learn fine. If the factor is 1.1 instead, you've exploded past any representable number, and the loss goes to infinity on step one. The whole game is engineering that factor to be one, on average, in both directions at once.



Corn

So it affects convergence speed, gradient scale, final quality. And it's the word "randomised" that Daniel is poking at, because some of these parameters are not random at all.



Herman

Right — you have parameters sampled from a distribution, parameters set to fixed deterministic values, and parameters deliberately set to a specific non-zero number because someone found it keeps the network alive. "Randomised weights" flattens all three into one phrase.



Corn

Then let's do the tour. The schemes with names.



Herman

The classic lineage starts with LeCun, late nineties. Variance of one over fan-in, where fan-in is the number of inputs to a neuron. The goal is to preserve the variance of activations as they pass forward, so what goes into layer ten has roughly the same energy as what went into layer two. Then Xavier, also called Glorot, which is a compromise: preserve activation variance going forward and gradient variance coming backward. The uniform version samples from plus or minus the square root of six over the sum of fan-in and fan-out. Nice detail: when fan-in equals fan-out, Xavier collapses back to LeCun. They're the same scheme at a square layer.



Corn

One family, two settings.



Herman

Then the punchline of the whole lineage: He, also called Kaiming. Variance of two over fan-in. That factor of two exists because ReLU zeroes out half the activations. Half your signal is gone by construction, so you scale up to compensate. And He initialisation exists as a named method because Glorot performs poorly with ReLU. It wasn't an aesthetic upgrade — the previous default broke on the new activation function.



Corn

A scheme for every activation. Almost like the activation function is the actual customer.



Herman

It is, and that's the thread I'd pull all episode. These aren't interchangeable fashions. Xavier assumes a roughly linear regime. He assumes ReLU. Give tanh the same treatment and you get saturation problems, and there's recent work — a fixed-point analysis of tanh of ax, ICLR, looking for values of a that keep the network off the flat part of the curve — that outperforms Xavier on robustness, data efficiency and convergence speed, with or without normalisation. Each activation earns its own arithmetic.



Corn

Hold on. That's not just "we tuned the number." That's the same derivation, redone for a different curve.



Herman

Redone, yes. Same logic, different mathematics. And the practical takeaway for anyone in the audience actually writing an init function is simple: you don't pick a scheme because a tutorial told you to, you pick it by asking what your activation does to the signal, and then running the variance arithmetic backwards from that.



Corn

Beyond the big three, the plan says you brought friends.



Herman

Several. Orthogonal initialisation — semi-orthogonal matrices with an activation-dependent scaling, and the elegant property that in a deep linear network initialised this way, training time to convergence is independent of depth. Unitary initialisation for LSTMs, which improves long-sequence modelling because the weights stay unitary throughout training. Fixup, for residual networks without batch norm, which is a whole recipe: zero-init the classification layer and the last layer of each residual branch, He elsewhere, depth-dependent scaling inside the branches, scalar multipliers at one, biases at zero. T-Fixup is the Transformer version, for transformers without layer norm.



Corn

A recipe, not a distribution.



Herman

It's a recipe because the disease is structural. Residual connections stack additively, so the variance of the residual stream grows with depth, and Fixup is a set of counter-scalings to hold it down. Say a bit more about what "grows with depth" means, because it's the same arithmetic as before wearing a different hat. Each residual block adds its output to the stream rather than replacing it. If every block contributes variance of one, the stream after N blocks has variance N. A hundred-layer residual network starts life with a signal a hundred times too hot, and normalisation layers are exactly the thing you removed when you chose Fixup. So instead you shrink each block's contribution — that's the depth-dependent scaling, roughly one over the number of layers, applied inside the branches.



Corn

So Fixup is Xavier's job done by hand, layer by layer, because you fired the automatic system.



Herman

That's a fair compression, yes. LSUV goes a different way entirely — it's data-dependent. Orthonormal initialisation, then a forward pass on a single random minibatch, layer by layer, dividing each layer by the standard deviation of its own output until every layer's output variance is about one. It looks at real data before the real training starts. And sparse initialisation gives a small subset of weights larger values and leaves the rest at zero, holding total variance around one over fan-in.



Corn

So within the "random" family we have uniform, normal, orthogonal, data-dependent, sparse, and recipe-based. What about the biases? Everyone forgets the biases.



Herman

You would ask about the biases.



Corn

I've been waiting all episode. There's a small chance you'll defend them.



Herman

Biases are usually zero, and usually is the interesting word. In multiplicative units — the forget gate of an LSTM — the bias gets initialised to one. Not random, one. Because a zero-biased forget gate starts mostly closed, and you want the gradient to flow through the gate at step one. ReLU neurons sometimes get a small positive bias, around 0.1, so the neuron isn't already dead on arrival — dying ReLU is permanent, once a ReLU's output sits at zero it can stop receiving gradient entirely. There's even an RNN convention from Le, Jaitly and Hinton of initialising the recurrent weights to the identity matrix, which is the closest thing to residual connections avant la lettre.



Corn

The identity matrix. The most aggressive possible reading of "randomised weights."



Herman

And LayerNorm parameters — weights at one, biases at zero. Every single time, deterministic. Nobody rolls dice for those. If you audited a modern checkpoint and sorted the parameters into "sampled from a distribution" versus "set by hand," the hand would win on sheer count of parameter classes, even where the sampled tensors win on raw numbers. The deterministic choices are sprinkled through every architecture like punctuation.



Corn

Okay. Textbook tour done. Now the part I actually enjoy: the canonical from-scratch language model ignores all of it.



Herman

NanoGPT, Karpathy's reference implementation. The initialisation function does two things. Every Linear and every Embedding weight gets a normal with mean zero and standard deviation 0.02. Every bias gets zeros. That's it. No Xavier, no He, no fan-in anywhere. Then a second pass, architecture-specific, over one class of parameter: every residual output projection gets re-initialised with the base std divided by the square root of two times the number of layers. For a twelve-layer model, 0.02 over the square root of twenty-four, which is about four and a half thousandths, roughly a fifth of the base std.



Hilbert

Four point one.



Herman

Four point one, thank you, four point one. Roughly a fifth.



Corn

He's got the desk calculator again.



Herman

The comment in the code is one line: apply special scaled init to the residual projections, per the GPT-2 paper. That single line is doing the job Fixup and T-Fixup do with a full recipe — holding residual-stream variance steady as depth stacks up. One line, no ceremony. And there's a lovely detail at the end: nanoGPT ties the token embedding and the language-model head. They are literally the same tensor. One initialisation serves both, which is the most efficient kind of convention — the one you only pay for once.



Corn

A flat 0.02 for everything, then one corrective tap on the way out.



Herman

That's the whole thing. And it's not reckless — there's a sweep from last year that took ReLU MLPs and GPT-2-style transformers and scanned the initial standard deviation logarithmically, mapping where gradients vanish and where they explode, and found a broad stability band between one hundredth and one tenth. 0.02 sits comfortably inside it.



Corn

So the number everyone treats as folklore is folklore with a laboratory alibi.



Herman

The alibi came later than the folklore, which is how a lot of this field works, but yes.



Corn

But wait — why 0.02 at all? If the stability band is that wide, why does the number look so specific? Why not 0.05, why not 0.01?



Herman

Because the band is wide but not free. The sweep found vanishing on one end and explosion on the other, and 0.02 is comfortably interior — close enough to the middle that you're not flirting with either cliff, small enough that the tied embedding and the head start life well-behaved. And once GPT-2 shipped with it and the model worked, every descendant inherited the number for free. Nobody re-derived it; they copied a repo. The honest genealogy of 0.02 is less "derived from first principles" and more "survived contact with a large training run, then reproduced."



Corn

Which is also how most recipes in cooking work, for what it's worth.

L

Larry

Friends, are your weights too randomised? Then you need the Glorot Gladstone, the only garden ornament engineered with Xavier fan-in fan-out variance, so every neuron in your shrubbery learns a different feature instead of all sulking in symmetry. It starts at four thousand and ninety nine dollars, but while I talk the price is dropping, now three thousand, now two thousand, now one thousand and one, and I am throwing in a depth-scaled residual birdbath, a forget-gate bias, a spare activation function for ReLU zeroing, and a limited edition orthogonal hedge trimmer.

L

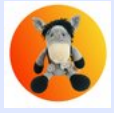
Larry

That is nine hundred, eight hundred, seven hundred, and a data-dependent LSUV watering can, six hundred, five hundred, four hundred, three hundred, two hundred, one hundred, and, oh, the deal has ended before I finished describing it. The Glorot Gladstone, because convergence speed is a garden you can hear.



Corn

Let it go.



Herman

I'm not letting it go, because the hedge trimmer claim is wrong. Orthogonal initialisation is not a trimming method, it's a constraint on the singular values of the weight matrix — the whole point is that the matrix is semi-orthogonal, so it can't shear the activation space. If you trim a hedge orthogonally you just get a box. There's no variance



Herman (continued)

preservation in a box, Corn. None.



Corn

It's a hedge trimmer, Herman.



Herman

The forget-gate bias I'll allow. The birdbath I'll allow. But that trimmer is going in the show notes with a correction.



Corn

Where were we. Right — that's the language-model convention. Now the ASR side.



Herman

Whisper, as the reference point, and the first thing to say is that it's not one convention, it's several, and they're all architectural. You've got a Conv1d front-end — kernel size three, the second convolution strided by two, GELU activations — taking mel spectrograms. Then the audio encoder's positional embedding is a fixed sinusoidal buffer, computed once and registered with register_buffer. Deterministic. Not learned, not sampled. The text decoder's positional embedding, in the same checkpoint, is a learned parameter, an empty tensor of context length by width, randomly initialised. Two philosophies of "where am I in the sequence," cohabiting in one model.



Corn

One half of the model gets a map drawn by a draftsman; the other half gets an etch-a-sketch.



Herman

And it makes sense per half, which is the maddening part. The audio encoder's positions are fixed physical frame positions — a second of audio is always the same number of frames, so the sinusoids never need to learn anything. The text decoder's positions live in



Herman (continued)

a symbolic space where position meaning is learned alongside everything else. The convention follows the data, not the task label.



Corn

But how does that play out in practice? If the sinusoids are fixed, does the encoder lose anything — is there some position information it can't represent?



Herman

In principle, any fixed basis is a constraint — the model can't reshape its notion of position even if the data wanted it to. In practice, for audio, the frame clock really is uniform and physical, so the constraint costs approximately nothing. The interesting contrast is that this is the exact opposite bet from the decoder half, where position is treated as fully learnable because it's fully symbolic. Same file, same authors, opposite conventions — chosen by asking what the input actually is, not what the model is for.



Corn

What about the attention itself?



Herman

Q, K, V and output as separate Linear layers, and the key projection is built without a bias — Linear of width to width, bias false. A concrete little architectural choice that has nothing to do with any random distribution. Now, the honest caveat. Whisper's Linear and Conv1d subclasses override only the forward pass, for dtype casting — they don't define a bespoke initialiser in the model file, so they inherit PyTorch's defaults. But I have not seen the explicit init code with my own eyes. The file's long, the relevant part never made it in front of me, and I'm not going to sit here and assert Whisper uses Xavier or Whisper uses defaults without checking. That's a claim I want to verify before I make it.



Corn

So the picture there is honest uncertainty, not a blank.



Herman

Which is itself informative. Nobody's blog post leads with Whisper's init function. The convention that matters for ASR isn't in that file at all.



Corn

Because of what you're about to say about checkpoints.



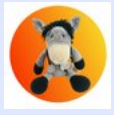
Herman

Because the dominant modern ASR practice is not to start from random weights, period. There's an ASRU paper — Preformer, sequence-to-sequence — where the encoder is initialised with a pretrained wav2vec 2.0 acoustic model and the decoder with a pretrained DistilGPT2. They hit 4.6% character error rate on AISHELL-1, a 27% relative reduction over a vanilla hybrid CTC-attention Transformer trained the honest way. And an Interspeech paper on speech intent classification that initialises a Conformer-Transformer with an encoder pretrained on speech recognition — 90% intent accuracy on SLURP — and finds ASR pretraining much more effective than self-supervised pretraining for that downstream task. The initialisation, in both cases, is a checkpoint. Someone else already paid for the randomness.



Corn

And it's worth spelling out what that means in engineering terms. You're not loading a distribution and sampling — you're loading a file, a state dict, megabytes of someone else's converged training run, and your "initialisation" is a load_state_dict call.



Herman

The randomness happened once, upstream, in a data centre, on somebody else's budget. And notice the Preformer detail — the two halves of that model come from different lineages entirely. The acoustic half descends from speech self-supervision, the text half from GPT-2. The initialisation isn't one convention; it's a graft. Two pretrained trees, spliced at the cross-attention. That's the modern shape of "how do I start training."



Corn

Daniel's browsing, then. What's he actually looking at on the Hub?



Herman

Fine-tunes and format conversions, almost exclusively. The top ASR downloads: a WhisperKit CoreML conversion repo at ninety-one million downloads a month, a faster-whisper-small quantisation at six point six million, then the base OpenAI checkpoints themselves — whisper-base at seventy-three million parameters, whisper-small at two hundred forty-two million, the large-v3-turbo at eight hundred nine million. Every one of them is either a pretrained checkpoint or derived from one. Not a single random-init artefact in the whole top shelf.



Corn

Ninety-one million downloads of something nobody initialised.



Herman

The download numbers are people loading weights that were random once, in a data centre, years ago. So when Daniel sees different conventions for different model families on Hugging Face, my read is that he's seeing different pretraining lineages more than different distributions. The ASR-specific bits are architectural — the convolutional front-end, the fixed sinusoidal positions, cross-attention between audio and text. There is no named ASR initialisation scheme in the literature I've seen. The closest thing to a task-specific convention is "initialise from wav2vec 2.0."



Corn

Which would make the honest answer to his question: same kind of randomness at the pretraining stage, and then no randomness at all at the stage everyone actually runs.



Herman

That's a nice compression and I think it holds. The random-init question is real — but it's the question you ask about the pretraining stage, not about the fine-tune. Today's version of that point is just that for ASR, even the pretraining often isn't from scratch — the encoder starts life borrowed.



Corn

Corn, your turn to be wrong at me. What's your instinct on why the field settled this way?



Herman

I just had mine.



Corn

You did, and you were coherent, so now I'm owed. Here's my suspicion, and you tell me if it's lazy: the textbook schemes come from a world of MLPs and convnets, trained by grad students on a GPU each. Transformers with LayerNorm arrived and quietly removed most of the reason those schemes existed — the normalisation keeps the scales honest regardless — and the field just never went back to rewrite the syllabus. So nanoGPT's flat 0.02 isn't heresy, it's what you do when the safety net is already installed.



Herman

Partly. LayerNorm does a lot of the variance control that Xavier and He were invented to provide, and that's exactly why the residual-projection scaling is the only special case nanoGPT bothers with — the one place where depth still accumulates additively and normalisation alone doesn't fix the variance of what gets added. But I wouldn't say the schemes became irrelevant. Same sweep paper: Kaiming fan-in converges faster and more stably than Xavier under ReLU, which is the variance-preserving theory still earning its keep in the plain networks. And they tracked the layer-by-layer variance of the Q, K and V weights through a twelve-layer pretraining run — the variances equilibrate into narrow bands, shallow layers expanding fast, deep layers drifting slowly. The network rearranges the initialisation into its own preferred shape within training. The starting point still matters; it just matters less than the textbooks imply and more than the folklore admits.



Corn

"More than the folklore admits" — what's the worst case, concretely?



Herman

Dying ReLU is the gruesome one. Improper initialisation can permanently inactivate neurons, and the deeper the network, the worse the gradient instability gets. The Stiefel manifold work from this year is blunt about it: He, Xavier and orthogonal don't necessarily regulate the pre-activation mean or control activation sparsity, and their effectiveness often diminishes in very deep architectures. You don't get a slower model. You get a smaller one — neurons that never woke up.



Corn

A building where some floors were never wired.



Herman

And nobody tells you which. The diagnostic is nasty too — a network with dead neurons still trains, still produces a loss curve that goes down, still ships. The damage is invisible unless you go looking at per-unit statistics mid-training, which almost nobody does. You'd only notice that the model is a bit worse than it should be, and there are a hundred other suspects before "my init quietly killed eight percent of layer fourteen."



Corn

Which is the most unsettling kind of bug — one that presents as a slightly disappointing model.



Herman

There's also an older, stubbornly practical objection: Koturwar and Merchant, back in 2018, arguing that He and Xavier work well on the standard benchmark datasets and not on more practical ones, because they ignore the data statistics entirely. Their other observation I love, from the same paper: very few researchers try to train their models from scratch at all. That was true then, it's truer now, and it explains the Hugging Face shelf better than any theory does.



Corn

Which raises the question of whether random init is even the right frame going forward. Is anyone attacking that?



Herman

There's a February paper that does, and it's a strange one. Instead of directly training the network from random weights, they first pre-train it to fit random noise, which leads to a structured and non-random parameter configuration — and then training converges faster, with no extra data. It's most effective for implicit neural representations, those coordinate-network models where the network is the image, but the principle is the thing: replace the coin toss with a cheap rehearsal. Randomness becomes a starting problem you solve before the real problem.



Corn

Pre-warming the engine by driving it around an empty car park.



Herman

The counterpoint keeps the field honest, though. Someone tested whether Xavier, He and orthogonal help variational quantum circuits escape barren plateaus — the quantum analogue of vanishing gradients — and the benefits were marginal. These heuristics grew out of classical architectures, and they don't automatically travel.



Corn

So the hierarchy is: architecture and normalisation decide the method, the method is often a plain number that somebody's grandmother chose, and in practice most people outsource the whole question to whoever pretrained the checkpoint.



Hilbert

My brother-in-law Ron set up band PA systems in the mid eighties. Six nights a week, halls in Haringey and one pub in Muswell Hill that had a compressor he disliked on principle. Before every job he zeroed the desk. Every fader down, every gain to the stop, every EQ flat. Took him twenty minutes with a torch in his teeth. The other engineers set the desk to roughly where last night's job had been and adjusted by ear. Ron said starting from zero was the only honest way to know what the room had done to you. I helped him carry the flight cases, which is how I know the Haringey licensing man, a Mr. Doleman, took eleven weeks to issue the noise-limiter certificate the hall needed, and that Ron priced a limiter of his own at three hundred and forty pounds from a shop on Gray's Inn Road and never bought it. He marked the desk settings for each hall in a exercise book. He gave me the book when he gave it up. It's in the drawer under the kettle.



Corn

The pub compressor. What did it do to him?



Hilbert

Nothing to him. It pumped on the snare. He said you could hear it breathing and refused to route the drum bus through it, so the pub got the drums acoustic, and they liked it better.



Herman

The zeroing is the interesting part for me. He was paying a twenty-minute tax every night for a guarantee the other engineers got approximately for free by remembering yesterday.



Hilbert

They didn't get it free. They got it quickly and wrong. Ron's way you find out what the room did. Their way you find out what the last room did. He was slower on the first song and faster on the rest, and he never once had to apologise to a singer.



Herman

So here's the technical question his book raises for me, and it goes back to Corn's stack of identical amplifiers: the whole point of the layered initialisation we described is that one value, applied at every layer, is wrong in a way that compounds. Ron was working at one layer. If you'd stacked his desks, the honest-zero approach would have needed a per-floor correction.



Hilbert

Herman, you've taken twenty minutes to say the first song. The correction I'd want to know about is your scaling one, the fifth-of-the-base. If you had a hundred of those channels, all settable to one number and one depth correction on the way out — I'm asking whether that works. Whether the whole stack comes up balanced.



Corn

Balanced for what?



Hilbert

For what I'd do with it.



Herman

I'll tell you what doesn't work — the hundred identical settings with no correction. That's the additive variance problem exactly. With the depth correction, in the idealised case, the claim holds up: the analysis says variance stays bounded layer to layer. Real hardware drifts, and a real desk at unity is not unity two weeks later. Ron's exercise book was the recalibration.



Hilbert

Then the book was the useful part. He kept it very neat for a man who drank that much. Never missed a hall, never missed a date.



Corn

Of course he didn't.



Herman

Ron's method survives in miniature, actually — that noise pre-training paper is a twenty-minute zeroing ritual for a network, on the theory that a known starting point beats a remembered one.



Corn

Ninety-one million downloads a month and the whole field's real starting point is one guy's exercise book. Which is where I wanted to land the episode anyway: how much of the initialisation literature is actually about the models people run? The top of the ASR charts is one pretrained family, converted and quantised a hundred ways, and the canonical from-scratch LLM uses a flat 0.02 plus one depth correction instead of any named scheme. The textbook machinery is for models nobody trains anymore.



Herman

The honest version of that is that the machinery was always about the pretraining stage, and the pretraining stage is where the money is — it's just that almost nobody's in that room. When someone is, the conventions they use turn out to be architectural: nanoGPT's residual scaling, Whisper's fixed sinusoids, Fixup for the normalisation-less. The task label never appears in the init function.



Corn

No. The task turns up in the fine-tune, which has no dice in it at all. That's the thing to hand Daniel: the randomness he's looking for on Hugging Face happened years ago, in one data centre, and the shelf is the descendants.



Herman

My weird prompts dot com has the full episode archive, by the way, if you want the earlier train-from-scratch episodes alongside this one. Thanks to Hilbert Flumingtop for producing, as ever, and for correcting my arithmetic before I'd finished the sentence.



Corn

For more along these lines, there's episode three, Safetensors or something else. This has been My Weird Prompts. If the show earns a review from you, leave one — it's the cheapest fine-tune we get.



Herman

See you tomorrow, everyone.



Corn

See you tomorrow.

SOURCES

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

1. **karpathy/nanoGPT, master branch, retrieved 2026-11-21. [primary]**
https://raw.githubusercontent.com/karpathy/nanoGPT/master/model.py

2. **openai/whisper, main branch, retrieved 2026-11-21 (truncated at ~8,000 chars; init code not reached). [primary]**
https://raw.githubusercontent.com/openai/whisper/main/whisper/model.py

3. **Hugging Face Hub, created 2022-09-26, retrieved 2026-11-21. [primary]**
https://huggingface.co/openai/whisper-base

4. **Hugging Face Hub, created 2024-10-01, retrieved 2026-11-21. [primary]**
https://huggingface.co/openai/whisper-large-v3-turbo

5. **Hugging Face Hub, created 2024-02-28, retrieved 2026-11-21. [primary]**
https://huggingface.co/argmaxinc/whisperkit-coreml

6. **Wikipedia, Weight initialization, last edited 2026-08-10, retrieved 2026-11-21.**
https://en.wikipedia.org/wiki/Weight_initialization

7. **Wikipedia, Vanishing gradient problem, last edited 2026-08-29, retrieved 2026-11-21.**
https://en.wikipedia.org/wiki/Vanishing_gradient_problem

8. **Han, Weight Initialization and Variance Dynamics in Deep Neural Networks and Large Language Models, submitted 2025-10-10.**
https://arxiv.org/abs/2510.09423v1

9. **Lee, Kim & Choi, Optimized Weight Initialization on the Stiefel Manifold for Deep ReLU Neural Networks, revised 2026-10-20.**
https://arxiv.org/abs/2509.00362v2

10. **Lee, Choi & Kim, Robust Weight Initialization for Tanh Neural Networks with Fixed Point Analysis, ICLR 2025, revised 2025-03-02.**
https://arxiv.org/abs/2410.02242v2

11. **Krishna Kumar, On weight initialization in deep neural networks, revised 2017-05-02.**
https://arxiv.org/abs/1704.08863v2

12. **Koturwar & Merchant, Weight Initialization of Deep Neural Networks(DNNs) using Data Statistics, revised 2018-03-17.**
https://arxiv.org/abs/1710.10570v2

13. **Wang & Alkhalifah, Target noise: A pre-training based neural network initialization for efficient high resolution learning, submitted 2026-02-06.**
https://arxiv.org/abs/2602.06585v1

14. **Peng et al., Can Classical Initialization Help Variational Quantum Circuits Escape the Barren Plateau?, submitted 2025-08-25.**
https://arxiv.org/abs/2508.18497v1

15. **Deng et al., Improving Hybrid CTC/Attention End-to-end Speech Recognition with Pretrained Acoustic and Language Model, ASRU 2021.**
https://arxiv.org/abs/2112.07254v1

16. **Huang, Balam & Ginsburg, Leveraging Pretrained ASR Encoders for Effective and Efficient End-to-End Speech Intent Classification and Slot Filling, Interspeech 2023.**
https://arxiv.org/abs/2307.07057v1