Episode Book

EPISODE #6016

What 70 Billion Parameters Actually Means

PUBLISHED	15 October 2026 at 20:37 UTC
DURATION	24:06
LISTEN	myweirdprompts.com/episode/what-parameters-actually-mean-74125/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.5
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Seventy billion parameters gets repeated on every earnings call, but almost nobody who says it could tell you what one parameter is. This episode builds the artificial neuron from the ground up — weights, bias, activation — then works through the math objects underneath it: scalars, vectors, matrices, and tensors. Along the way, the conversation lands on why stacked linear layers collapse into a single matrix without a nonlinearity, why the word "rank" means three unrelated things depending on who's talking, and which parts of the biological neuron the artificial one actually kept. It closes by stress-testing the metaphor against a real transformer, where the perceptron equation survives intact but the perceptron as a unit of meaning does not. By the end, that number with a B on the end feels like something you could almost picture.

TRANSCRIPT



Corn

Seventy billion parameters. The number gets said on every earnings call and every podcast intro, and almost nobody who repeats it could tell you what one parameter actually is.



Herman

That's the whole gap, isn't it. We use the vocabulary constantly and mean almost nothing by it.



Corn

Daniel's asking us to fix that. He wants the building blocks, ground up. Start with the single artificial neuron, then the objects underneath it, scalars and vectors and matrices and tensors, then how those map onto learned parameters, and then the hard question at the end, whether the neuron analogy even describes a transformer anymore.



Herman

And that last part is where it gets contested. It's not a rhetorical flourish.



Corn

So the arc is, build the neuron, count the parameters, then stress-test the metaphor against what's actually inside a real model.



Herman

Let's build it.



Herman

The vocabulary of this field is borrowed from two places that don't talk to each other. The neuron comes from nineteen-fifties neuroscience. The matrix and the tensor come from nineteenth-century linear algebra.



Corn

And the borrowings have drifted since.



Herman

Both of them. The math drifted less, honestly. A matrix in PyTorch is a matrix the way Cayley would have written it. The neuron drifted a lot more. Here's the one-sentence version of the whole episode. The perceptron equation survived into the transformer completely intact. The perceptron as a unit of meaning did not survive at all.



Corn

Those are two different claims and they're both true.



Herman

That's the tension. So we make three moves. We build the neuron from scratch. We build the math objects and connect them to the parameter count. Then we stress-test the neuron against a transformer that's sitting in front of us.



Corn

And the payoff by the end is that seventy billion parameters feels like a countable thing you could almost picture, rather than a marketing number with a B on the end. Let's do it.



Herman

The canonical artificial neuron takes n inputs, each one multiplied by a weight, sums all of those products, adds a bias term, and passes the result through an activation function. In notation, z equals w transpose x plus b, and then y equals g of z.



Corn

That's the entire thing.



Herman

The MIT computer vision course calls the w transpose x plus b part the linear layer, and g the activation function. Those two names are worth keeping.



Corn

Unpack the pieces for me. Go slowly, because this is the load-bearing part.



Herman

The weight scales an input. It's how much this particular input matters to this particular neuron. If the weight is large and positive, that input pushes the neuron hard toward firing. If it's near zero, the neuron barely listens to that input at all. If it's negative, the input suppresses the output.



Corn

So a weight is an importance score, essentially.



Herman

An importance score that training sets. The bias is different. The bias shifts the threshold. It's how eager the neuron is to fire regardless of what it's being shown. A large positive bias means this neuron is trigger happy, it'll fire even on weak evidence. A large negative bias means it takes a lot to wake it up.



Corn

And the activation function.



Herman

The activation is what makes the whole thing nonlinear. This is the piece people skip past and it's the piece that does the work. Chollet's argument is the cleanest version of why. If you stack Dense layers with no activation between them, the whole stack collapses to a single affine transform. Algebraically, applying one affine map and then another gives you back an affine map, with the matrices multiplied together and the biases recombined.



Herman (continued)

Two layers of weights with nothing between them is just one layer of weights wearing a coat.



Corn

So depth is only meaningful because of the nonlinearity.



Herman

Chollet's line is that the deep network would be a linear model in disguise. That's the sentence to remember. Every architectural claim you've ever heard about depth buying you representational power is downstream of that one fact. Take away the activation and you have a very expensive way of doing least squares.



Corn

Which explains why nobody's ever shipped a deep network with the activations stripped out.



Herman

And why people keep rediscovering it by forgetting to add one.



Corn

Give me the history, briefly. Rosenblatt, right?



Herman

Nineteen fifty-seven. Rosenblatt's perceptron used a step function. Output one if z is greater than zero, output zero otherwise. It fires or it doesn't, no in-between. And a lecture deck from IIT calls it, I'm quoting, a very crude biological model, one that performs linear classification.



Corn

Crude is doing some work in that sentence.



Herman

Crude is the honest word. Modern neurons use smooth or rectifying activations. ReLU is max of x and zero, which is about as simple as nonlinearity gets. Softmax at the output when you want a probability distribution. The step function only survives in textbooks as the starting point.



Corn

Now the part Daniel actually asked about. Which piece corresponds to the biology.



Herman

The most direct analog to the weight is synaptic connection strength. How strongly one cell talks to the next. That mapping is good. The bias maps onto a firing threshold. One source writes it as x-sub-zero equals one, w-sub-zero equals beta equals minus theta, where beta is the bias and theta is the activation threshold. Once you see that, the bias stops looking like an accounting convenience and starts looking like the resting potential of a cell.



Corn

And the activation function.



Herman

Loose. Very loose. Which brings me to the honest part of this. A paper in Neuron in twenty nineteen describes artificial networks as an extreme abstraction of biological ones, and extreme is the right word. Biological signal is discrete spikes in continuous time. The artificial neuron gets a single real number per forward pass. A biological weight is a dynamic stochastic chemical synapse. The artificial weight is one fixed scalar that gets updated by a training rule. Biological learning is local and online and neuromodulated. Ours is global, batched, and exact.



Corn

No dendrites in the model.



Herman

No dendrites, no cell body, no axon, no subtypes. Real neurons come in dozens of types and are modulated by a whole menagerie of factors the artificial neuron doesn't even have a field for.



Corn

So the neuron is a computational description, not a biological one.



Herman

It tells you what arithmetic happens. It does not tell you what a brain does. I spent a long career around the second kind of claim, and the two get blurred constantly.



Corn

Hang on, before you move on. You said stacked Dense layers with no activation collapse to one layer. Does that mean you could train a no-activation network and then just fold it down to a single matrix afterwards?



Herman

You could fold it down, yes. That's exactly what the algebra says. And the folded version would be cheaper and produce identical output. The collapse isn't an approximation, it's an identity.



Corn

So anybody who's ever trained a deep stack of linear layers has been paying for depth they didn't receive.



Herman

That's a sharp way to put it. They paid for depth and received a single matrix.

L

Larry

Hi there, friend. Do you find yourself with more neural parameters than you know what to do with? Then you need ParameterBox, the only monthly subscription service that ships a curated selection of tensors directly to your door.

L

Larry

Our bronze tier gets you one scalar per month, carefully hand-selected for the discerning learner. Silver gets you a vector, arranged in order, I'm told that matters. Gold gets you a matrix, and our platinum tier, which I recommend, ships an entire rank-three tensor in a foam-lined crate. Two hundred and fifty-six axes of pure potential. Cancel anytime, which we define as never, and our retention team will find you.



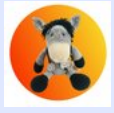
Herman

That's not what a rank-three tensor is. That's not even close. Rank three means three axes. Two hundred fifty-six axes is...



Corn

Herman.



Herman

A crate of two hundred and fifty-six axes isn't a tensor, it's a filing cabinet, and the bronze tier is selling a single number as a subscription, which is...



Corn

Let it go.



Herman

Fine. It's a filing cabinet.



Corn

So that's the arithmetic of one neuron. But we've been saying weight and matrix and tensor almost interchangeably for twenty minutes. Let's actually pin those down, because the muddle is real.



Herman

Goodfellow's definitions are the cleanest. A scalar is just a single number. A vector is an array of numbers arranged in order. A matrix is a two-dimensional array of numbers, so each element is identified by two indices. And a tensor is an array of numbers arranged on a regular grid with a variable number of axes.



Corn

That last one is doing an enormous amount of work in one sentence.



Herman

Chollet's framing is friendlier. A tensor is a container for data. Usually numerical data. Rank zero is a scalar. Rank one is a vector. Rank two is a matrix. Rank three is a cube of numbers. And it keeps going. A tensor is defined by three attributes. The number of axes, the shape, which is a tuple of dimensions, and the dtype, which is what kind of number is inside it.



Corn

And the terminology trap here, which I've fallen into.



Herman

In code, call it ndim, not rank. Matrix rank means the number of linearly independent directions in a matrix, which is a completely different concept. If you say rank in a linear algebra class and rank in a PyTorch tutorial, you mean two unrelated things, and the



Herman (continued)

collision has cost people real time. There's also a third meaning, which is that in physics a tensor carries a coordinate-transformation property that a plain multidimensional array doesn't necessarily satisfy. Three meanings. One word.



Corn

Wonderful.



Herman

The conventions are worth knowing because they encode what the data is. Vector data is samples by features. Timeseries is samples by timesteps by features. Images are samples by height by width by channels. Video adds a frames axis in there. MNIST training images are a rank-three tensor of shape sixty thousand by twenty-eight by twenty-eight, dtype uint8.



Corn

Sixty thousand images, each one a twenty-eight by twenty-eight grid of pixel values.



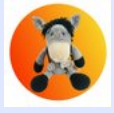
Herman

And no learned parameters in it at all. Which is the hook for the next part. Now that we have the objects, let's count them, because this is where seventy billion stops being a slogan.



Corn

Connect them first. Where does a matrix stop being a math object and start being a parameter?



Herman

A Dense layer is literally output equals relu of matmul of input and W, plus b. W is a matrix. b is a vector. Both are attributes of the layer. Both live on the layer object, both get saved when you save the model, both get loaded when you load it.



Corn

So weights are not always scalars.



Herman

Weights are almost never scalars at the layer level. A single layer's weights form a matrix. The individual scalar entries are the atomic learned values. Both statements are true at different zoom levels, which is why people talk past each other. When someone says a weight, singular, they mean one entry. When the same person says the weight matrix, they mean the whole layer. The parameter count counts the scalars.



Corn

And parameter versus weight versus bias.



Herman

Parameter is the umbrella term. Every weight is a parameter. Every bias is also a parameter. One writer puts it plainly. Each weight or parameter is a float that was tuned during training. The tuning is what makes it a parameter. Not the shape, not the name, just the fact that gradient descent moved it.



Corn

Then Daniel's other question. What about tensors that aren't parameters?



Herman

There are a lot of them. The input data tensor, the MNIST image tensor we just described, is a tensor and it's not a parameter. The intermediate activations inside the forward pass are tensors and they're not parameters. The residual stream in a transformer is a tensor and it's not a parameter.



Corn

So what's the dividing line.



Herman

Learned and stored, versus computed per forward pass. Parameters are the things that get written to disk and shipped. Everything else is arithmetic that happens and then evaporates.



Corn

Make seventy billion concrete. Actually do the arithmetic.



Herman

Take a transformer block. There are four attention weight matrices, for the query, key, value, and output projections. Each one is d-model times n-heads times d-head. In most architectures n-heads times d-head equals d-model, so each of those four is roughly d-model squared.



Corn

Four d-model squared per attention block.



Herman

Then the MLP. Two matrices, each d-model squared times four. The factor of four is architectural and basically every transformer since the original in twenty seventeen has gone with that ratio, four times the model width in the hidden layer. And then there's a wrinkle. Some modern LLMs use what's called a gating variant where the up-projection is split into two matrices that get multiplied together elementwise. That takes the MLP from two-D-F to three-D-F.



Corn

So every architectural fad you read about changes the count by a single-digit multiple.



Herman

That's exactly what it changes. Let's do embeddings, because that's the number that surprises people. Take a sixty-five thousand five hundred thirty-six token vocabulary and a model width of five hundred twelve. The token embedding table alone is five hundred thirty-six million eight hundred seventy thousand nine hundred twelve parameters. That is a lot of the model. Biases are approximately zero in count. Layernorms are d-model per block, which is noise.



Corn

So the count is dominated by d-model squared repeated per layer, plus the embedding table.



Herman

Which is why parameter count scales so predictably with width and depth. Double the width and the per-layer cost goes up four times. Add a layer and you add one more block of the same shape. Nobody's hand-tuning these numbers. They're a handful of repeated matrix shapes.



Corn

And a weight vector is then just a row of one of those matrices.



Herman

A row of a matrix, a column of another. The bias vectors are vectors. The attention projections are matrices. Nobody's lying when they call them by the array name. They're just describing shape instead of role.



Corn

The equation survived. The question is whether the metaphor did. And this is where it's contested.



Herman

Mechanically, the neuron is still there. Every MLP neuron in a transformer is still a weighted sum plus an activation. Nothing in the architecture repealed the perceptron. Where it fails is as a unit of interpretation. Anthropic's work on this, from October twenty twenty-three, says the most natural computational unit of a neural network, the neuron itself, turns out not to be a natural unit for human understanding, because many neurons are polysemantic. They respond to mixtures of seemingly unrelated inputs.



Corn

Give the examples, because the examples are what land.



Herman

In Inception v1, a vision model, a single neuron responds to faces of cats and fronts of cars. Two completely unrelated categories, one neuron. In Anthropic's small language model, a single neuron responds to a mixture of academic citations, English dialogue, HTTP requests, and Korean text.



Corn

That's four unrelated things. What's the hypothesized cause.



Herman

Superposition. Neural networks want to represent more independent features of the data than they have neurons, so they assign each feature its own linear combination of neurons. The representation is overcomplete. Their one-layer transformer had a five hundred twelve neuron MLP layer, and they found tens of thousands of features in it. They expanded the dictionary up to two hundred fifty-six times the neuron count, which is one hundred thirty-one thousand and seventy-two features.



Corn

Five hundred twelve neurons. One hundred thirty-one thousand features.



Herman

And this is where sparse autoencoders come in. Dictionary learning, essentially. They extract features that are more monosemantic, and those features turn out to be effectively invisible in the neuron basis. They found features for Arabic script, for Hebrew script, for DNA sequences, and for base64 strings, and none of those features corresponds to any single neuron. Activate the base64 feature and the model starts generating base64 text. There's no neuron whose job that is.



Corn

That's a clean result.



Herman

The Arabic feature is my favorite of them. Arabic script is about one tenth of one percent of the training tokens, and it made up eighty-one percent of the tokens where that feature fired. So it's a rare feature that fires almost exclusively when it's supposed to.



Corn

So the dictionary learning finds real structure that the neuron view misses.



Herman

And monosemantic neurons do exist in the wild. There's work that found clean examples of both monosemantic neurons and superposition, so it's not that the neuron is never a unit. It's that it's an unreliable unit. Sometimes it lines up with a feature and sometimes it's holding four features in a trench coat.



Corn

But there's a live disagreement here, and I don't think it gets flagged enough.



Herman

There is. A paper from this year argues that some of what looks like polysemanticity is actually a lexical confound. A neuron might fire for the word form itself, like bank, in both the river sense and the money sense, rather than compressing two unrelated concepts into one direction.



Corn

And that's a materially different story.



Herman

It's a very different story. Across models from a hundred ten million parameters up to seventy billion, the lexical-only condition consistently exceeds the semantic-only condition. And somewhere between eighteen and thirty-six percent of the sparse autoencoder features blend senses. Though the confound itself sits in under one percent of the activation dimensions.



Corn

So it's real but it's small, and it doesn't rescue the neuron.



Herman

Even if you clean out every lexical confound, the one-layer model with five hundred twelve neurons still holds tens of thousands of features. The compression is genuine. The question is just how much of the apparent mixture is compression and how much is the model noticing that the string b-a-n-k is the string b-a-n-k.



Corn

So the honest framing is that the equation survived and the metaphor didn't.



Herman

That's the whole episode in seven words.



Hilbert

I agree with you.



Corn

...With which part?



Hilbert

The metaphor being broken. The equation surviving. All of it. But you've got the wrong culprit. It's not the math. It's the naming. Somebody called it a neuron in nineteen fifty-seven and we've been living with that decision ever since.



Herman

Your objection is to the word.



Hilbert

My objection is to the word. I've been keeping a list.



Corn

A list.



Hilbert

Every time I hear tensor used wrong. Every time I hear parameter used as if it means weight. Every time somebody says the model thinks and means a matrix multiplication. I keep a running count. I've got a laminated card in my wallet. Fine-tip pen. I update it most weeks.



Herman

You carry a laminated card.



Hilbert

I had it laminated because it was getting worn. It's a two-sided card. And the lamination has started to delaminate along the top edge, so I've got a second card that's just the corrections to the first card. I keep that one in the other pocket so I don't have to get both out at the same time.



Corn

How many entries are on the card.



Hilbert

That's not something I'm going to get into.



Herman

Do you keep an index.



Hilbert

I keep an index. It's alphabetical. I'm not going to share the index.



Corn

Of course it's alphabetical.



Hilbert

I've been mailing photocopies to a professor. Not to the department. That goes to a building full of graduate students and nobody reads anything, I know how that works. I mail it to his home address. He knows what it is by now. I consider that peer review. That's the whole function of peer review as far as I can tell. You send a thing to a man's house and he either writes back or he doesn't.



Herman

Has he written back.



Hilbert

Not yet. But the card's had two revisions since I started. Anyway, it's the naming. That's the problem. You can't call a weighted sum a neuron and then act surprised when it doesn't behave like one.



Corn

Say more about the nephew thing. You said you had a relative doing this.



Hilbert

I'll leave it there.



Corn

Fine. Actually, wait, I hadn't thought about it this way. If the naming is the problem, that's an argument for renaming the unit, not for abandoning it.



Herman

Which would be a purely cosmetic fix, if the unit doesn't line up with anything.



Corn

Except the equation still describes what the thing computes. Every MLP neuron is a weighted sum plus an activation. That's not a naming problem, that's a fact.



Hilbert

That's fair. That's fair. But you'd still be calling it after the one thing it isn't.



Corn

The alternative is calling it a linear layer plus a nonlinearity, which is what the computer vision people already do.



Herman

That's the MIT naming, actually. They call the sum the linear layer and the function the activation. No neuron anywhere in it.



Hilbert

Which is what I've been saying. Just call it what it is. It's not complicated.



Corn

Then the stress test resolves cleanly, and it isn't a mystery. The perceptron equation went into the transformer intact. Every MLP neuron still computes a weighted sum and then an activation. What didn't survive is the neuron as a unit of meaning. The models hold more features than they have neurons, so the single neuron is often four things at once.



Herman

The disagreement about how much of that is genuine compression versus a lexical artifact is live right now, and it matters. If it's compression, we need to find the real units. If it's partly lexical, some of the mystery dissolves and interpretability gets easier.



Corn

Either way, the number is the thing I'll take from this. Seventy billion parameters is a handful of repeated matrix shapes. Four attention matrices of d-model squared, two MLP matrices of four d-model squared, repeated for every layer, plus an embedding table. You could nearly count it on a card.



Herman

A laminated one.



Corn

A laminated one, corrections in the other pocket. If you're enjoying the show, a review wherever you found us goes a long way.



Herman

Thanks to our producer, Hilbert Flumingtop.



Corn

If you enjoyed this one, try episode fourteen, AGI's Crossroads. This has been My Weird Prompts. Send us your own prompt on Telegram at t dot me slash MWP listener bot, and we'll see you soon.

SOURCES

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

1. **Anthropic, Towards Monosemanticity, published 2023-10-04 [primary]**
   https://transformer-circuits.pub/2023/monosemantic-features

2. **Goodfellow, Bengio, Courville, *Deep Learning* Ch. 2 [primary]**
   https://www.deeplearningbook.org/contents/linear_algebra.html

3. **François Chollet, *Deep Learning with Python* Ch. 2 [primary]**
   https://deeplearningwithpython.io/chapters/chapter02_mathematical-building-blocks/

4. **MIT *Foundations of Computer Vision*, perceptron definition**
   https://visionbook.mit.edu/neural_nets.html

5. **kipply, LLM Parameter Counting, 2022-03-30**
   https://kipply.github.io/blog/transformer-param-count/

6. **transformer parameter formulas**
   https://github.com/bencottier/ml-parameter-count/blob/main/parameter_count.py

7. **gating-einsum / MLP parameter note**
   https://github.com/jax-ml/scaling-book/blob/main/transformers.md

8. **Elhage et al., Toy Models of Superposition, 2022-09-21**
   https://arxiv.org/abs/2209.10652

9. **Finding Neurons in a Haystack, sparse probing**
   https://arxiv.org/abs/2305.01610

10. **Ait Hou & Hwa, Polysemanticity or Polysemy?, 2026-04-01**
    https://arxiv.org/abs/2604.00443

11. **biological vs. artificial neuron comparison table**
    https://mikenguyen13.github.io/ai_in_action/004-biological-vs-artificial-intelligence.html

12. **Neural networks for perceptual processing (weights as connection strengths)**
    https://pmc.ncbi.nlm.nih.gov/articles/PMC2323553/

13. **(19)31044-X**
    Neuron, Direct Fit to Nature, 2019
    https://www.cell.com/neuron/fulltext/S0896-6273

14. **tensor rank vs. ndim caution**
    https://shakeri-lab.github.io/dl-book/chapters/appendices/a2-tensors.html