**"Neural Networks: Logic, Not Magic"**

**I didn’t write this to teach. I wrote what I wish someone told me on day one.**

**Written by  
Aelrizon  
 GitHub:** [**github.com/Aelrizon**](https://github.com/Aelrizon) **Reddit: u/Aelrizon**

**Stage 1: Foundations — Thinking, Logic, and the Neuron**

Goal: Build a fundamental understanding of how both humans and machines “think.”

* What is logic, and how does it learn?
* The human brain as a learning model
* What is a neuron — simple and precise
* How a single neuron makes decisions
* Activation functions — no magic, just math
* Gradients explained logically, not mystically

### **Stage 2: Architecture — From Neuron to Network**

Goal: Explain how networks are built and how they think in layers.

* Why one neuron isn’t enough
* How a layer of neurons works
* What makes a network deep
* How data flows through the network
* Forward and backward — logic, not algorithms
* Loss and learning — simple error and correction
* Optimization — what it’s and why it matters

### **Stage 3: Control — Practical Intelligence**

Goal: Prepare the reader to build and understand networks independently.

* How to design your own neural network
* Why data matters more than architecture
* What overfitting is and how to avoid it
* Dropout, BatchNorm, and other "magic" made human
* How to write your own library — principles and structure
* Comparing with frameworks (PyTorch, TensorFlow)
* You’re ready — now go teach others

**Stage 1: What Is Logic, and How Does It Learn?**

Aristotle was the guy who figured out the truth before anyone else. He proved that nothing’s really complicated — if you apply logic.Logic is your brain’s weapon. With it, you can figure out anything.

That’s why I keep saying:Neural networks are logic — not magic.

Every invention ever? Built on logic.You use it every day — but the second something looks "hard", people act like logic just disappears.Before I explain anything to you — I want you to stop and try answering things yourself.Using logic.But for that —you’ve gotta know the rules.

### **1. Law of Identity**

If it says “glass” — it’s the thing you drink from.Don’t change meanings mid-conversation. Call things what they’re.People argue because they picture the same thing differently —they never agreed on what it meant.That makes the argument dumb.

### **2. Law of Non-Contradiction**

If everyone sees a blue poster,but one guy says it’s red —someone’s wrong. Both can’t be true.(That is — if you’re following the first law.)

### **3. Law of the Excluded Middle**

If one thing cancels out another — then one of them’s false.  
You can’t say “yes” and “no” at the same time.

There’s only one truth. It’s either true, or it’s not.

If you don’t get these three laws — you’re not learning. You’re just memorizing.And trust me — that’s not gonna take you far.

**Stage 1: The human brain as a learning model**

How does a human learn? I don’t know what your answer is — but I’ll tell you how it really works. Simple and straight.

When a child is born, they know nothing. They only have instincts, built by nature and God. They can’t speak — because speaking isn’t instinct, it’s a learned skill. And from the very beginning, they start learning.

We’re not talking about body growth or vocal cords here.  
We’re talking about how a child learns language.

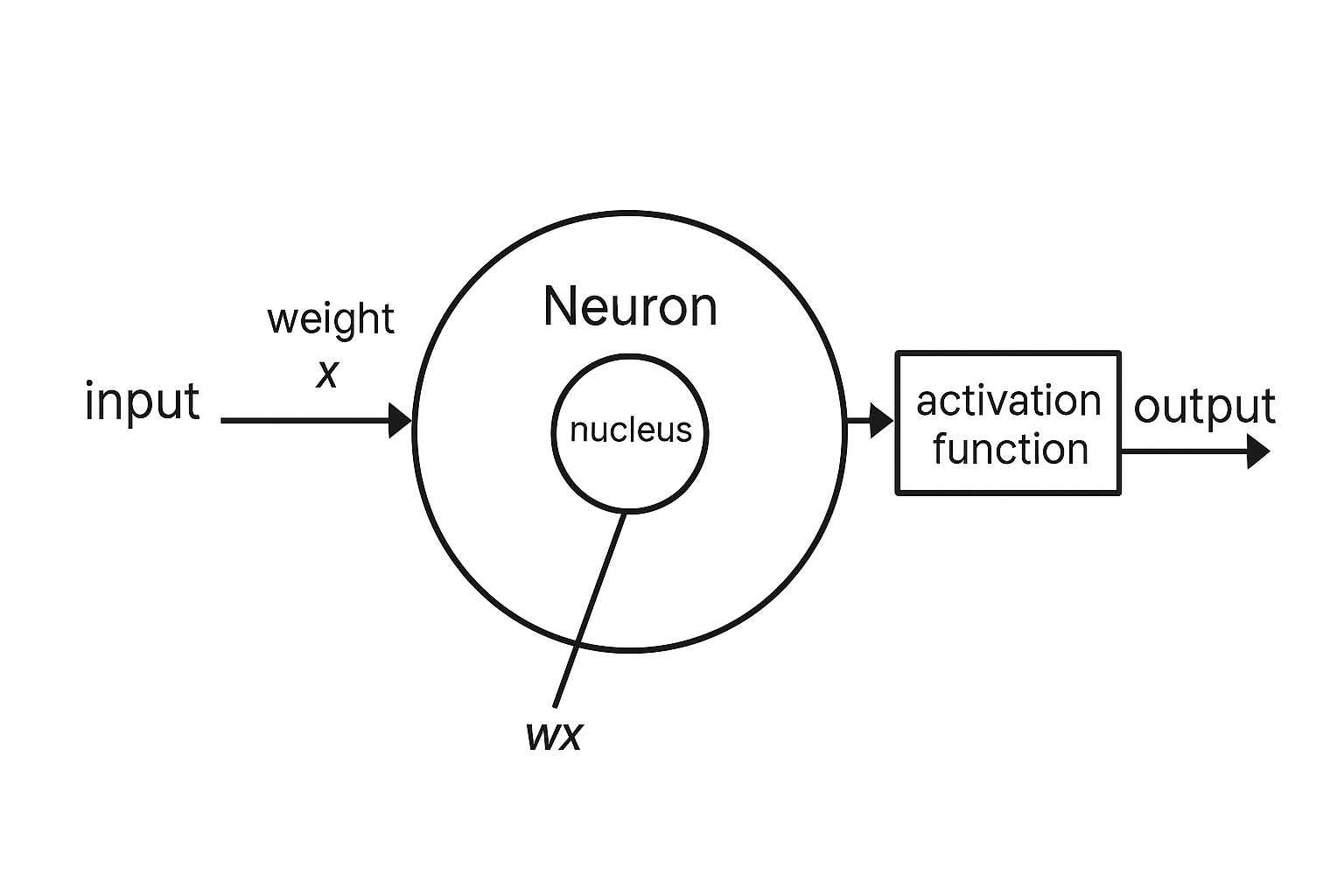
A child is thrown into a world they don’t understand at all.  
Parents show things, say words, do actions — but none of it makes sense to the child.You can’t explain grammar to a baby or make them memorize words like an adult — and they don’t need that.  
The brain adapts. It starts picking up patterns — which words come up, what they relate to, what reactions they cause.  
The child doesn’t wait for lessons — they learn on their own. That’s called unsupervised learning.

Neural networks can do the same. You throw data at them, and they figure it out themselves — just like a baby in a language environment.

But there’s another way. Say an adult starts learning Python. They read books, take courses, get the info. Then they try writing code. They make mistakes. A tutor or the terminal tells them where the error is.They fix it and remember. That’s supervised learning — when someone helps you spot mistakes and improve.

That’s why I say: if you don’t understand how a child learns — you don’t understand how your own brain works.  
 And if you want to build neural networks, start with yourself.

**Stage 1: What is a neuron — simple and precise**



So what, already forgot logic and basic math?A little pressure — and your brain taps out?

Come on, it’s simple.

This is how a mechanical neuron works. We’re not biologists — we don’t need to know how the real one works.But the mechanical one? You better understand that.A neuron receives information. It could be anything.Let’s say it’s a question sent to ChatGPT.  
Makes sense that the output should be an answer.

Then come the scary words — weights and activation function.  
People panic. Relax. It’s not magic.

Weights are ChatGPT’s knowledge.It just multiplies your input by what it knows.Sounds strange? Sure. But that’s how math works.  
The weights are its brain. They give meaning and weight to the input — that’s how it decides what to say.

Here’s the formula:  
wx = input × weight.

Next comes the activation function — the moment the neuron decides: Should I respond — or ignore this?If the signal is strong, the output is an answer.If it’s weak — it’s silence.Just logic. No magic, no chaos. A neural network isn’t a miracle. It’s a system.

Just remember: I’m not writing this because I have to — I’m writing it because I can.  
(Yeah, that was between the lines. If you got it, you got it.)

And if you're from a university — don’t worry.  
I'm still open to conversation. Just know: this time, it's on equal terms.

**Stage 1: How a single neuron makes decisions**

Activation threshold: how to stop a neuron from reacting to noise or nonsense?

Yeah, it’s time to answer that — because we’ve already talked about signal strength.Here’s how it works.

A single neuron is already a tiny neural network.  
Let’s say its mission is to detect whether there’s a woman in the picture.Yeah, tough task — even for me. That’s the world we live in.One neuron isn’t enough, but let’s pretend.

The output is simple: “Yes” or “No.”

Now we feed in a photo of a woman.The neuron checks facial features, body shape, other details.If it all clearly adds up — it confidently says: Yes.

So why do we need a threshold?

Without it, the neuron could fire just from a single detail.  
It might catch one soft feature — and scream “It’s a woman!”  
without looking at the rest of the image.That’s an error.

The threshold gives the neuron time to “think” —to wait for more evidence before making a decision.

Sure, no threshold means faster output.But it also means way less accuracy.It reacts to anything — even noise.

With a threshold: strong signal = “Yes.” Weak signal = “No.”

If you’re a neuron that activates from one strand of hair — you definitely need a threshold.  
And if you still can’t tell a man from a woman — don’t worry. You’re not alone.

**Stage 1: Activation functions — no magic, just math**

The decision is made. Now let’s figure out what inside the neuron pushes the button — "go" or "stop." That’s the activation function.

This is the final checkpoint before the output. A filter.  
Let’s say: if the image shows a woman — the output should be 1.  
If not — 0.

But the neuron itself might produce any number between 0 and 1 — say, 0.3.Then the activation function steps in and rounds it down to 0.The neuron doesn’t fire. Just zero.No signal — no woman.

But if it’s 1 — that’s a confident signal. It goes through.  
The neuron fires.That’s the whole point.

Someone decided to call this a “function.” Sure, why not.

And if you want the real formulas — fine.It’s called ReLU, or Sigmoid, or Tanh.hey all do the same thing — just with different moods.But that part’s for the nerds who don’t mind wasting a few hours.

And hey — if you still don’t get how this works, maybe your activation function is just stuck on zero.  
Relax. Happens even at MIT.

**Stage 1: Gradients explained logically, not mystically**

How many types of learning are there? Answer: two. If you didn’t get it — that’s sad.

With a teacher — gradient works.  
Without a teacher — gradient doesn’t.

In math, a gradient is just the fastest way to change the value of a function.Sounds scary? It’s not.Nothing complicated here.

Weights are the brain of a neural network. That’s where all the knowledge lives.At first, the network knows nothing.  
So the weights are picked randomly — just some number to multiply the input with.

And that’s when training begins.

You give it a picture of a woman.The network says: “There’s nothing here.”We say: “Wrong. There is.”

The gradient shows how the network’s knowledge needs to change.  
It remembers the picture — and adjusts the weights.

Picture a map with points from 0 to 100. The network lands on point 23.78.The gradient says: “No — it should be 23.23.” The network steps that way.

The gradient sets the target.The network moves to it.

If you didn’t get it — maybe you’re looking the wrong way.  
I’ve already moved on.

### **End of Stage 1**

Congrats. If you’ve made it this far — I didn’t break you. I tested you a few times, sure. Pressed a little. But if you’re still here — it means you didn’t just read. You thought.

Right now, you know more than 99% of people. Even the ones sitting in lectures, still scared of the word “neuron.” You just walked through what most people spend years trying to understand.

They get formulas.You got clarity.

They get assignments. You got logic.

They memorize definitions. You now explain things yourself. That’s a different level.

Most people think this stuff is for “smart people.” No. You just need to kill the fear and switch your brain on. You did. That’s why you’re ahead.

If someone asks how much you know — here’s the answer:  
Enough to teach the ones who tried to teach me.

And if you’re also tired of how this world works — you’re not alone.

If you’re sick of fancy words hiding simple truths — you’re not alone.

The world won’t get clearer on its own. Someone’s gotta say things like they are. Even if it’s just one voice at first. You just saw what that looks like. What you do with it now — that’s on you.

**Take a breath. Next up: Stage 2. Now we build the brain.**

**Stage 2: Why one neuron isn’t enough**

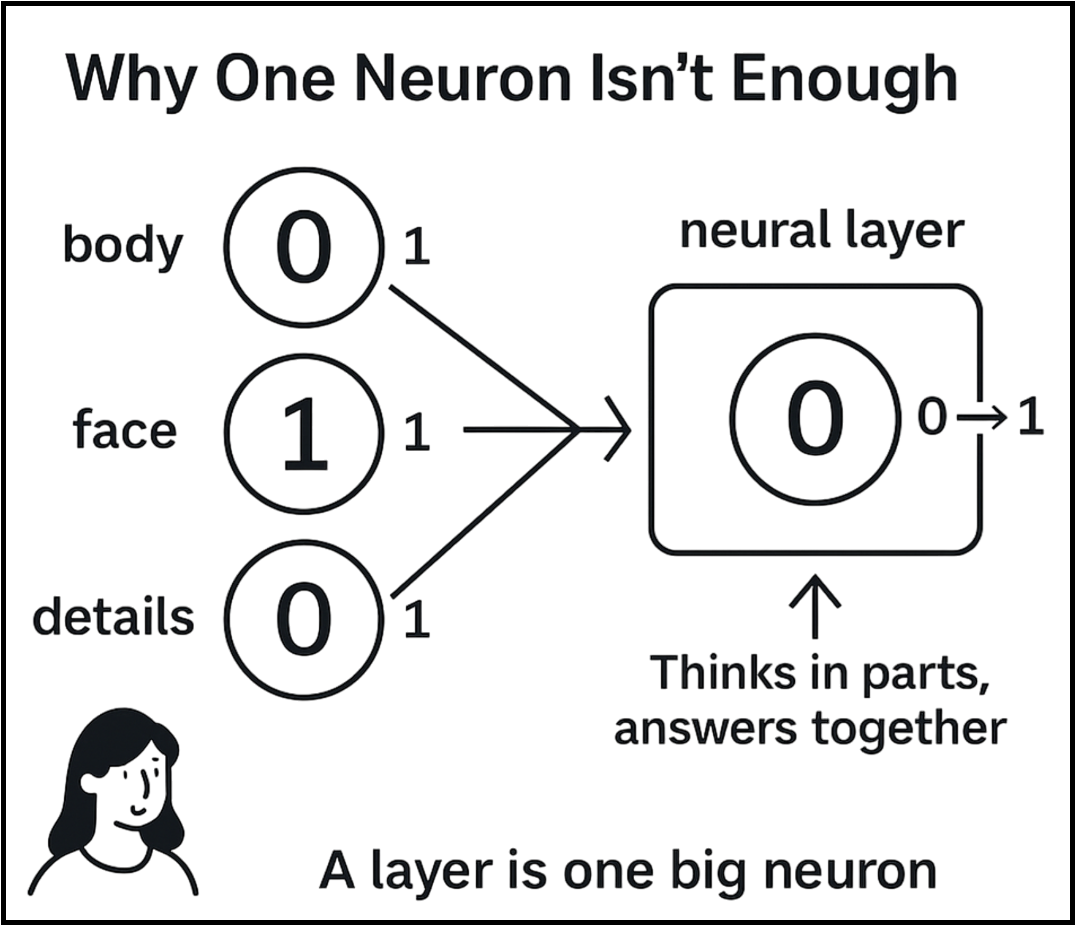
How do you solve a hard problem easily?  
Break it into smaller ones.

This works everywhere.That’s exactly what Frank Rosenblatt thought when he suggested:what if we just connect multiple neurons into one layer?

Back to the woman-in-photo example.What makes her recognizable? Body, clothes, face, details.Too much for one neuron. It can’t handle all that — it’s limited.

The solution?

We use multiple neurons. Each one looks at a piece.



— One checks: is there a body?   
— Another: is there a face?  
— Another: colors, posture, etc.

Each neuron gives a “yes” or “no.”  
Then we connect them into one layer.  
Add an activation function — to keep outputs clean.  
Now we have a system that thinks in parts, but answers together.

That’s a neural layer.  
And here’s the key — a layer is basically one big neuron, just smarter.

So why stop there?

We take layers, stack them, connect them — and now we’re building real neural networks. Now the brain’s getting bigger.

If one couldn't handle it — don't worry. The team’s got this.

**Stage 2:How a layer of neurons works**

A layer is one big neuron — with one logical twist:  
inside it, there’s a bunch of mini neural networks.

Each neuron is like a mini-network. It has its own weights and its own job.And when they’re all packed into a layer — you’ve got a bunch of neural nets running in parallel.

That’s the key thing you need to know.

I’ve already talked about breaking things into parts — each neuron looks for its own thing: shape, detail, color, direction, and so on.

Now it’s important to understand how they do it. All neurons work at the same time.Each one has its own weights. Makes sense —different jobs mean different ways of thinking.

Let’s say one neuron gives 0.3. The activation function says: too weak — turns it to 0. Another gives 1 — solid. The others give their own values too.

You get a list like this: [0, 1, 0, 0, 1]. That’s the output of the layer.

Then this list gets passed forward. And the next neurons decide what to do with it.

While some are busy building theories — I just showed you how it actually works.

**Stage 2:What makes a network deep**

“Depth” — yeah, nice word they picked. If I were a professor at some university, I’d just say: “It’s a neural network with a bunch of layers.” But how it actually works —he probably wouldn’t explain that.

So let’s think logically. Here’s why one layer isn’t enough — and how compute power comes into play.

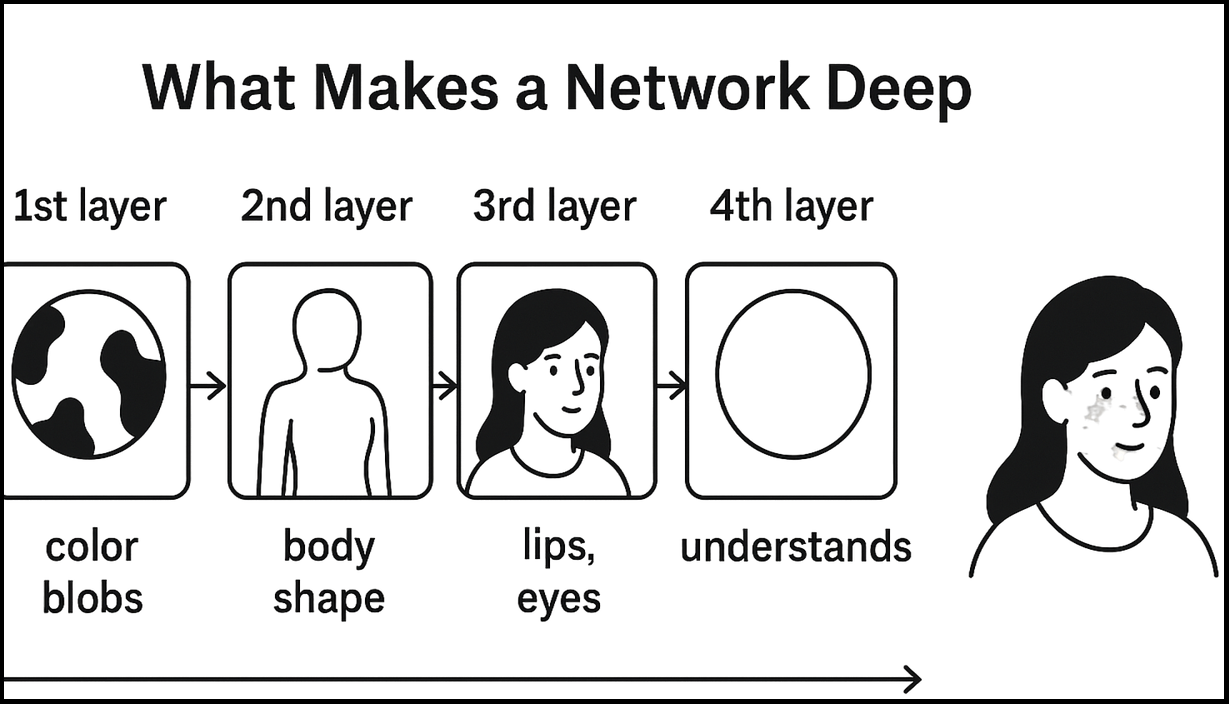
Back to the woman example.Yeah, she’s gonna follow us through this book. So what? Makes it easier. One layer — the network just stares at color blobs.What’s dark, what’s pink, where’s the edge.

Add a second layer — now it might catch the body, shape, structure.Third layer — it starts seeing a face, lips, eyes.  
Fourth — it begins to understand, not just guess.

That’s how depth works. Each layer is a step toward meaning.The network stops reacting — and starts thinking.

That’s what a deep network really is. Not “a lot of layers” — a shift toward understanding.

And honestly, I still don’t get why no one explains it like this.

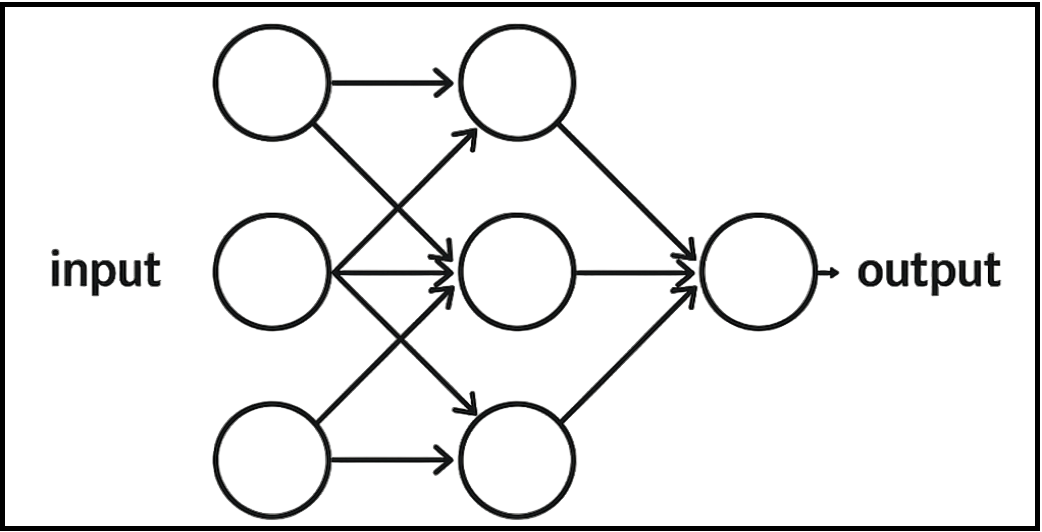


**Stage 2:How data flows through the network**

Inside a layer, all the neurons are connected.Each one is different, solving its own tiny task —  
but they work together like a team.

Every neuron gives a simple output: 0 or 1. A clear answer, based on a narrow job.And that’s exactly why it’s accurate.

All neurons inside a layer are wired to talk to each other — passing signals, nudging, adjusting.



But the layers themselves are also connected. How? Simple:Each layer has one input and one output. And the output of one becomes the input of the next.

That’s it.  
For one layer — it’s an output.  
For the next — it’s already an input.

The signal moves.The information flows through the structure.And the network does what it was built to do:it learns to think.

If you’ve made it this far — you’re starting to see how the machine thinks. Stick around. Next, you’ll see how it learns.

**Stage 2:Forward and backward — logic, not algorithms**

Alright. Time to talk about how learning actually works.No advanced math. No black magic. Just logic.Let’s say we built a neural network to find women in images.(Yeah, she’s staying with us till the end of the book. Get used to it.)

The network gets an input. It runs it forward through the layers. Filters it. Processes it.  
And then says: “Nope, nothing here.”  
But the truth is — there was.That’s a mistake.

Now what?

The network compares its answer with the real one. Sees the mismatch.And starts moving backward — layer by layer — whispering:Wrong answer. Adjust the weights.

But wait. How do the weights know which way to go?

Enter the gradient. I already mentioned it. Now it shows up and does its job — figuring out how the weights should shift to make the next answer better.

This feedback loop can repeat over and over.As long as you feed it examples with the correct answers,the gradient keeps teaching. The network keeps learning.

Mistakes aren’t the end.

**Stage 2:Loss and learning — simple error and correction**

“The only one who never makes mistakes is the one who does nothing.”(Einstein. Or someone equally smart. Doesn’t matter — the point stands.)

Same goes for neural networks.They won’t learn to recognize anything until they screw up a bunch of times and start to spot the patterns.

**Question:** Why does a neural network need to make mistakes?  
**Answer:** Because in the beginning, the weights are just random numbers.They mean nothing.And only through mistakes does the network start to understand which weights actually work.

And if you didn’t answer that yourself — I honestly don’t know how you made it this far.

“Loss” isn’t failure. It’s just a difference.The network says 0.9 — the real answer was 1.  
That’s it. That’s all it means.

Mistakes aren’t failure. They’re tools.  
Without them, the network has no idea where to go next.  
It’s not stumbling — it’s collecting data to get smarter.

Mistakes don’t break us.  
They build us.

**Stage 2:Optimization — what it’s and why it matters**

We already figured out how the network handles errors.We know where it needs to go to fix them.But here’s the real question — how does it move?

That’s what optimization is for. It decides what to do with the weights — how to actually change the numbers.

Should the network move carefully?Quickly?Should it remember past mistakes? Or just charge forward and hope for the best?

Without optimization, the network starts to wobble.Too fast, too slow, constantly overshooting or hesitating.It knows where, but doesn’t understand how. And that means it’s not learning —it’s just wasting time.

Optimization turns guessing into real progress.Not “close enough” — but “getting closer.”

In modern frameworks like PyTorch or TensorFlow,  
 this job is handled by tools like:

* SGD (Stochastic Gradient Descent) — simple, raw, effective
* Adam — smart, with memory
* RMSprop — a balance of speed and control

You’ll meet all of them.But for now — focus on how they think.Typing the code is the easy part.

Optimization isn’t just math.  
 It’s how the network stops wandering —  
 and starts walking like it means it.

### **End of Stage 2**

**That’s it. Stage 2 — done.**

Now you don’t just know how a neural network is built —  
 you get why it works the way it does.

You’ve seen how it forms, thinks, learns.  
 No fluff. No formulas. Just the truth.

You’re not just reading anymore.  
 You’re already thinking in networks.  
 You’re not waiting to be taught.  
 You could teach someone else — right now.

**And if you’re from a top university —  
 read carefully.  
 We’re not even at the end yet.  
 And it already hits harder than your intro lectures.**

**Stage 3: How to design your own neural network**

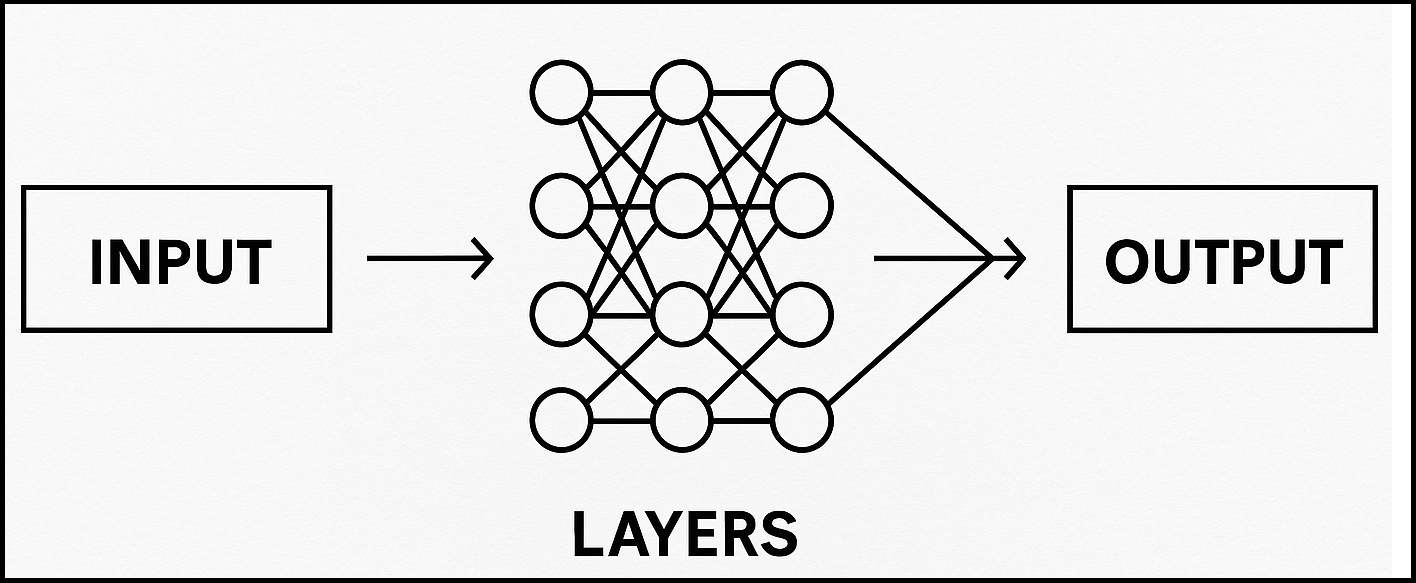
Building a neural network isn’t just about stacking layers.

It starts with an idea — and a plan to make it real.Writing code or collecting data? Easy.But designing the logic? That’s the real work.

Ask yourself the right questions:

* What problem is this network solving?
* What kind of data do I need? How much?
* What’s going in?
* What should come out?
* How many steps between input and result?

It’s not about how many layers or neurons.It’s about architecture.



You know the input.You know the output.  
Everything in between? That’s your call.

And that’s all you need to stop guessing and start building.

And no, ChatGPT can't design it for you. Yet.

**Stage 3: Why data matters more than architecture**

Data is the one thing a neural network can’t learn without.  
You can’t hand it a book called “How to trade” and expect it to start doing it.It doesn’t understand words.It learns through examples — and only examples.

You can spend weeks obsessing over architecture. But without good data? Even god-tier code won’t help. Architecture is just a tool. Nothing more.

If you want results, here’s the deal:

* Clear, properly labeled data
* As few errors as possible
* As much diversity as you can get

Let’s go back to the model that’s spotting women:

* If all the women in your dataset are wearing dresses → the model will decide: “Dress = Woman”
* If all the women are from one race → the model learns: “Race = Woman”

It doesn’t think like a human. It doesn’t question logic. It just memorizes whatever you feed it.

The computer is the dumbest thing in the room.  
 But at least it’s consistent.

**Stage 3: What overfitting is and how to avoid it**

A perfect result isn’t always a good sign.If a neural network scores 100% on training data — there’s a real chance it understood nothing.

That’s called overfitting.It just memorized everything like a burnt-out student who learned the formulas but has no clue what to do with them.

Here’s a classic:

You give the model 100 photos of women — and in every single one, they’re standing in front of flower-patterned wallpaper.The model learns: “flowers = woman.”

Then you show it a woman standing in front of bricks.And it says: “not a woman.”

It’s not dumb.It just memorized, instead of understanding.That’s what overfitting really is.Now, the good news: this isn’t some unsolvable mystery.We’ve got tools — and they work.

The most effective ones:

* Validation set — test on new data, always
* Dropout — randomly “turn off” some neurons during training
* Batch Normalization — helps stabilize the learning
* Early stopping — stop training when things stop improving
* Data augmentation — shake up the inputs (rotate, shift, blur, etc.)

Don’t make your network a straight-A student.  
Make it someone who actually gets it.

**Stage 3: Dropout, BatchNorm, and other "magic" m.human**

All these tools exist for one reason:To help the model learn better and not lose its damn mind halfway through training.

Back then, people had to do everything by hand.Now we just use tools that work.Here are the ones that actually matter:

#### Dropout

A model should use all its neurons,  
not just rely on a few overachievers hiding in the middle layer.If it leans too hard on the same neurons — it collapses.

Dropout says: “Take a break.”It temporarily shuts off random neurons,forcing the network to think differently and share the load.

Result? More flexibility. Better accuracy. Stronger brain.

#### Batch Normalization

Sometimes, inputs fly all over the place.High. Low. Chaotic. Confusing.The network panics.

BatchNorm calms it down.It stabilizes the inputs — so every neuron can breathe.Think of it as deep breathing for AI:  
clean signals, faster learning, less stress.

#### Other tricks worth knowing:

* Weight Decay — don’t let your weights grow like weeds
* Learning Rate Scheduling — start fast, finish smooth
* Early Stopping — when it’s done learning, stop. Simple.

Don’t call this magic.  
It’s just basic tools to keep your model from losing its mind.

**Stage 3: How to write your own library — principles and structure**

Honestly, this chapter is more for me than for the reader.Most people can just use PyTorch or TensorFlow and live happily ever after.But I had to include it — because building your own library?That’s master-level stuff.And yeah, I’ll be writing mine soon.At this point, with everything you’ve learned —you can definitely figure out the structure.Sure, coding it is hard.  
But logically building the blueprint? Easy.

Here’s how it all starts:

#### **Neuron**

Input → multiplied by weight → activation → output.  
Clean and simple.

#### **Layer**

A class that holds a bunch of neurons.  
Feeds them all the same input, collects their outputs.  
That’s it.

#### **Network**

Stacks layers one after another.  
Can do forward pass. Can learn through feedback.  
It’s the brain.

#### **Loss Function**

Compares prediction to the right answer.  
Tells the network, “You messed up. Try again.”

#### **Backpropagation + Gradient**

Error moves backward.  
Gradient says which weights to move, and how.  
Boom — the model learns.

#### **Optimizer**

Decides how to move the weights.  
Fast? Careful? With momentum? Your choice.

And that’s it.  
 You build that — you’ve got your own library.

Not copied. Not borrowed.  
Yours.

**Stage 3: Comparing with frameworks (PyTorch,TensorFlow)**

You could’ve just learned PyTorch.You could’ve followed tutorials, line by line.But you chose to understand how this all works.And now — you’ve got an edge.

Frameworks like PyTorch and TensorFlow?They’re fast tools — with everything built-in.  
Layers. Neurons. Gradients. Backprop.Everything you’ve learned — it’s already there.

But here’s the difference:  
You’re not just calling model.fit()  
You know what’s happening under the hood.  
You’re not afraid to tweak architectures.  
You know what it means to “move the weights.”

These frameworks run on the same logic you already mastered.  
Now you don’t just use them —you can think like them.And if needed?You could build your own.

That’s why you’re not dependent.

**Stage 3: You’re ready — now go teach others**

You didn’t just finish this book.  
You understood how neural networks work.  
You didn’t copy.  
You built the brain from scratch.

Now you’re not just using models —  
you can build your own.  
You know how the machine thinks.  
And more importantly —  
you know how to think for yourself.

Most people study for years and never reach this point.  
You did it in a few dozen pages.

**From the author**

**So yeah — this is where we part ways.  
Now it’s on you:  
Some will forget everything.  
Some will grow — in work, in study, in how they think.  
Both are fine.**

**I’m good with either outcome.Because I didn’t write this to help people.I wrote it for myself.**

**I needed to lay out what I understood —Clearly. Logically. Humanly.**

**This book wasn’t posted for clout or praise.It’s here because I believe:Neural networks aren’t magic.And humans shouldn’t be black boxes either.**

**God gave us a mind —and it’s meant to work.The brain is a weapon.  
The only question is: will you choose to draw it?**

**Today, there’s universities, companies, billions of dollars — all circling AI.And yet, many who should explain it… stay silent.  
Or worse — act like it’s “too hard.”**

**I chose to speak differently.Now it’s your turn.**

**And before I go — you should know the truth behind all this:**

**I wrote this entire book — in just three days.Not because I rushed. But because when you really understand something —it writes itself.**

**This book isn’t the end — it’s your launchpad.  
If you want to explore more (with formulas, theory, and code), check these out:**

* [**Zero to Hero by Andrej Karpathy (YouTube series)**](https://www.youtube.com/watch?v=VMj-3S1tku0) **A step-by-step, hands-on course by one of the top minds in AI. Beginner-friendly but deep.**
* [**Deep Learning Specialization by Andrew Ng (Coursera)**](https://www.coursera.org/specializations/deep-learning) **The most famous ML course online. Great if you want structure, projects, and certificates.**

**This book was written with the support of Jesus Christ.  
 Thank you. Glory to Him.  
Forever and always. Amen.**

**Written by  
Aelrizon  
 GitHub:** [**github.com/Aelrizon**](https://github.com/Aelrizon)