Coding Nexus · Follow publication

Member-only story

Nested Learning in Machine Learning: A new ML paradigm for continual learning

5 min read · 12 hours ago

Algo Insights Following

Listen Share More

One of the strangest aspects of training machine learning models is how forgetful they can be.
You can spend weeks training a model on one task, and the moment you fine-tune it for something new… poof, it forgets the old one.

Humans don't do that.
You don't forget how to ride a bike just because you learn a new phone number.

That gap — between how we learn and how our models learn — is what **Nested Learning** aims to bridge.

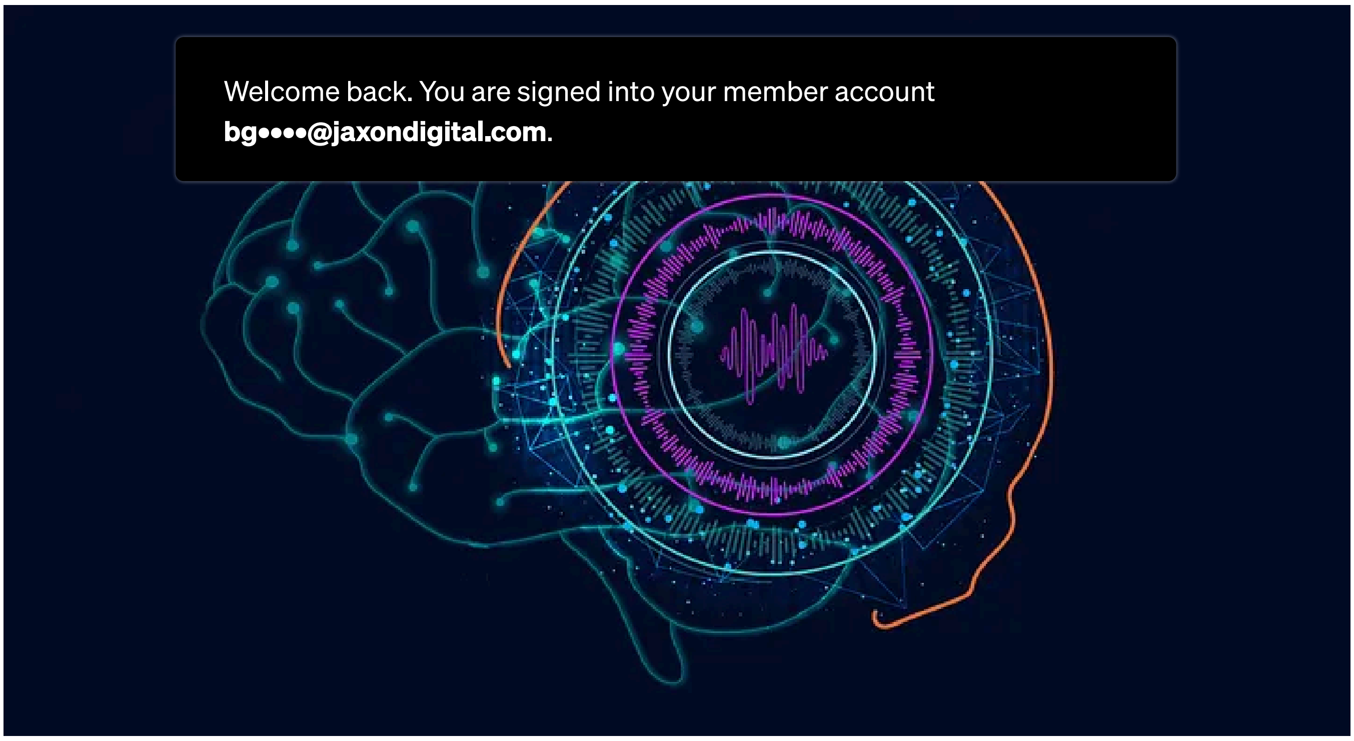

Image by — Google

. . .

Why Forgetting Happens

Let's start with the obvious: current models aren't built to *remember*.

They store everything in a large mix of parameters. So when you train them on new data, it's like scribbling over old notes in the same notebook. Eventually, all you have is noise — no apparent trace of what came before.

This problem has a name: **catastrophic forgetting**.

Researchers have attempted to fix it with various tricks — better optimizers, architectural tweaks, replay buffers — but the fundamental problem is more complex. It's that we've considered two things as separate when they probably aren't.

1. The **model architecture** (how it's built).
2. The **optimizer** (how it learns).

Nested Learning asks: what if they're actually the same thing?

. . .

Thinking

Here's the

A neural network isn't just *one* learner — it's a collection of smaller learners, each operating at its own pace.

Imagine a team where everyone learns in their own unique way.
One person picks things up quickly but forgets them by the weekend. Another takes longer but never forgets.
Together, they cover each other's weaknesses.

That's how Nested Learning views a model: as a collection of **nested optimization loops**, each with its own *update frequency*.

Some loops learn fast (short-term).
Some learn slowly (long-term).
And they all feed into each other.

Here's a small piece of Python-style pseudocode that illustrates the idea:

```
for layer in model.layers:
    if step % layer.update_rate == 0:
        loss = layer.learn(batch)
        layer.optimizer.step(loss)
```

Each layer (or component) updates at a different rhythm.
That rhythm allows the entire system to learn new things *without losing* previous knowledge.

Comparison of performance on language modeling (perplexity; left) and common-sense reasoning (accuracy; right) tasks across different architectures: Hope, Titans, Samba, and a baseline Transformer.

. . .

From Brains to Code

If this feels familiar, it's because the brain operates in a similar manner.
Neurons do not all fire or adapt at the same rate. Some parts of your brain change quickly when you learn something new; others only adjust slowly over time.

This multi-speed adaptation enables humans to learn continuously.
Nested Learning attempts to adopt that idea.

. . .

The "Continuum Memory" Concept

In standard Transformers, you can roughly think of:

- **Attention layers** act as short-term memory (what's happening right now in the sequence).

- **Feedforward layers** act as long-term memory (the general knowledge baked in from p

Nested Learning expands this concept into a **spectrum of memories** — short-term, mid-term, long-term — each updating at its own pace.

Something like this:

```
memory_levels = [
    MemoryModule(update_rate=1),   # short-term
    MemoryModule(update_rate=10),  # mid-term
    MemoryModule(update_rate=100)  # long-term
]

for step, data in enumerate(stream):
    for mem in memory_levels:
        if step % mem.update_rate == 0:
            mem.learn(data)
```

Each memory module handles a different timescale.
Together, they create what researchers call a **Continuum Memory System (CMS)** — essentially, a more flexible way for a model to retain both old and new knowledge.

. . .

Meet "Hope": The Self-Modifying Model

To test this entire theory, the team behind Nested Learning developed a model called **Hope** — short for "Hierarchically Optimized Predictive Engine" (yes, that's a mouthful).

Hope is built on the Titans' architecture, which already had a clever way of prioritising memories based on "surprise." But Hope goes further: it can actually *modify its own learning rules* as it runs.

Think of it as a Python script that modifies its own training loop during training.

```
def self_modify(optimizer):
    optimizer.momentum *= 0.95
```

```
        optimizer.learning_rate += random.uniform(-0.001, 0.001)

    while t
        los
        optimizer.step(loss)
        if step % 100 == 0:
            self_modify(optimizer)
```

This means Hope isn't just learning patterns in data — it's also learning how to *improve its learning*.
That's a significant advancement toward what people refer to as **meta-learning** or "learning to learn."

. . .

So, Does It Work?

Well, actually.

In tests on language modeling, reasoning, and long-context tasks (like those "needle in a haystack" benchmarks), Hope outperformed models such as Transformers, Mamba2, and even Titans.

It had lower perplexity (resulting in more sensible text), higher accuracy, and significantly improved memory management for long sequences.

In other words, it didn't panic when you threw it a 50,000-token input. It calmly kept track of what mattered and what didn't — something current models struggle with badly.

. . .

Why This Could Matter a Lot

If Nested Learning truly holds up, it could revolutionize how we train models. Instead of retraining large networks every time we want them to learn something new, we could allow them to **learn continuously** — maintaining existing skills while acquiring new ones on the go.

Imagine an AI that learns just like you do.

Not in a sin

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

. . .

The Bigger Picture

The authors of the paper — *"Nested Learning: The Illusion of Deep Learning Architectures,"* presented at NeurIPS 2025 — make a bold claim: What we've been calling "deep" learning might actually be "nested" learning all along.

In other words, maybe neural networks don't become smarter by *adding more layers* — they become smarter by nesting learning processes that work on different timescales.

That's quite a mind-bender. But it makes sense. Real intelligence isn't about depth. It's about *continuity* — the ability to link yesterday's lessons to today's problems.

. . .

Final Thoughts

Nested Learning is more than a simple architectural change. It offers a new perspective on what "learning" truly signifies for machines.

Suppose models can eventually handle their own memories and update themselves at different speeds. In that case, we might see AI systems that don't require constant retraining — they'll *keep learning,* just like humans do.

And honestly, that's pretty much the dream, isn't it?

AI Ai Agent Machine Learning Ai Tools Llm

Follow

Published in Coding Nexus

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

8.4K followers

Coding Nexus is a community of developers, tech enthusiasts, and aspiring coders. Whether you're exploring the depths of Python, diving into data science, mastering web development, or staying updated on the latest trends in AI, Coding Nexus has something for you.

Following

Written by Algo Insights

3.5K followers · 6 following

Algo Insights: Unlocking the Future of Trading with Code, Data, and Intelligence.
https://linktr.ee/Algo_Insights

No responses yet

Bgerby

What are your thoughts?

More from Algo Insights and Coding Nexus

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

In Coding Nexus by Algo Insights

How to Build Your Own AI Rig for Running Local LLMs (Gemma, Mistral, Qwen, GPT-OSS and Llama)

About three months ago, I realised I was utterly dependent on companies that didn't care about anything except power, Money, and control.

Oct 8 362 14

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expe er AI
wrapper, I th

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

Oct 23

In Coding Nexus by Civil Learning

The Guy Who Let ChatGPT Trade for Him — and Somehow It Worked

You know how everyone says, "Don't let AI touch your Money"? Well, someone on Reddit decided to ignore that.

Oct 8 274 8

In Coding Nexus by Algo Insights

4 Open-S

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

I’ve been co ne same for years. Editors, browsers, frameworks. Just...

Sep 3 453 9

See all from Algo Insights

See all from Coding Nexus

Recommended from Medium

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

In Level Up Coding by Fareed Khan

Building a Training Architecture for Self-Improving AI Agents

RL Algorithms, Policy Modeling, Distributed Training and more.

5d ago 751 16

In AI Software Engineer by Joe Njenga

Anthropic Just Solved AI Agent Bloat — 150K Tokens Down to 2K (Code Execution With MCP)

Anthropic just released smartest way to build scalable AI agents, cutting token use by 98%, shift from tool calling to MCP code execution

3d ago

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

In Coding Nexus by Algo Insights

Data Agents: The Next AI Revolution in How We Work With Data

I kept noticing the term Data Agent appear in AI papers lately. At first, I thought it was just another buzzword—like “copilot” or “AI…

5d ago 61 1

In Towards AI by Rohan Mistry

You're Not an AI Engineer. You're Just Using ChatGPT

What actual

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

4d ago 290 9

In Artificial Corner by The PyCoach

If You Only Read A Few Books About AI, Read These

Books recommendation

4d ago 326 3

In AI & Analytics Diaries by Analyst Uttam

Welcome back. You are signed into your member account **bg••••@jaxondigital.com**.

7 AI Agen

Last year, I u unting for insights that didn't always feel...

5d ago 74 3

See more recommendations