Not All Weights Are Equal

A heterogeneous design for a reasoning-first model

Benjamin Cowherd

Orbits Systems

2026

Abstract

We ask a simple question: does a dedicated reasoning component, given its own architecture and weight structure, outperform a standard homogeneous model of the same size on logic problems that require genuine generalization? In controlled experiments — Project Synapse — we isolated a ternary-weight reasoning module and tested it against an equivalent-parameter MLP baseline on multi-step logic problems using entity names never seen during training. The reasoning module outperformed the baseline by 12.2%. The result was consistent across multiple runs and reasoning types. We argue this improvement is explained by architectural specialization, not parameter count — and that it has implications for how AI systems should be designed.

1. Introduction

The field of AI figured out something powerful: scale a model up large enough, and reasoning just clicks. It emerges. That works. But it works the same way brute force works — it gets the job done without asking whether there is a better way.

The brain does not work by scaling one thing up. The region responsible for reasoning has a different structure than the region responsible for language. Reasoning areas show higher synaptic density and more complex connectivity than language production areas. The brain solved cognition by making different things different — allocating structure where structure is needed.

Current AI largely ignores this. A large-scale model uses the same weight structure for everything: remembering facts, forming sentences, solving logic problems, writing code. One painting. But cognition is not one painting. It is a puzzle — and the pieces are not the same shape.

This paper describes two of those pieces.

The argument is straightforward. Language can be handled by a relatively compact model — the structure of language is well understood. Reasoning is different. Reasoning needs density, iteration, memory, and discrete structure. Give reasoning its own component, built for what reasoning actually is, and it gets better at reasoning without making the whole system bigger.

Not all weights are equal. Reasoning and language are different computational processes. They need different substrates. Building one model that handles both with the same weights is a design choice, not a necessity — and it may not be the right one.

We propose a design with two parallel components. A ternary-weight reasoning component — weights constrained to {-1, 0, +1} — handles logical inference through iterative steps over explicit memory cells. A smaller float16 language component handles encoding and decoding natural language. The two communicate bidirectionally: language feeds semantic context into reasoning, reasoning feeds decisions back to guide generation.

To test the core claim before building the full system, we ran a controlled experiment called Project Synapse, isolating the reasoning component against a standard baseline. The reasoning component improved accuracy on novel multi-step logic problems by 12.2% at comparable parameter count. The results are reproducible. That experiment is the foundation this paper is built on.

This is Orbits Systems research, conducted by one person. The goal is not to beat existing large models at their own game. The goal is to find a better design principle — and we have early evidence for one.

2. Background

2.1 Scaling and Its Limits

The dominant approach in AI has been scale: more parameters, more data, more compute. This has driven the field toward increasingly large homogeneous models, and it has produced remarkable results. But scaling a uniform architecture answers the question of how much, not the question of what kind. The experiments in this paper ask the latter question.

Some large-scale models use a Mixture of Experts (MoE) architecture, routing inputs to different sub-networks for efficiency. This introduces a form of specialization. However, each expert in an MoE system uses the same weight type and the same basic structure — the specialization is in routing, not in the fundamental nature of the weights. Our proposal goes further: different cognitive functions should use structurally different weight representations entirely.

2.2 Reasoning as a Separate Function

Wei et al. (2022) showed that prompting models to produce intermediate reasoning steps significantly improves performance on complex tasks. This is consistent with the core claim here: reasoning benefits from explicit intermediate computation. But chain-of-thought is a prompting technique. We propose making the intermediate computation structural — built into the design.

2.3 Data Quality

Gunasekar et al. (2023) demonstrated that carefully curated, high-quality training data produces models that significantly outperform same-size models trained on larger but noisier datasets. This informs our training philosophy: the reasoning component trains on structured logic puzzles with verified correct answers, not general web text.

2.4 Ternary Weights

Ternary weight networks (Li et al., 2016; Zhu et al., 2016) have been studied primarily as a compression technique for inference efficiency. Our use of ternary weights is motivated differently. Logical inference is fundamentally discrete — a conclusion either follows from premises or it does not. Ternary weights are a structural prior that matches the weight representation to the nature of the computation. They are not a compression choice. They are a design choice.

3. Project Synapse: Proof of Concept

The experiment was designed to answer one question cleanly: does a dedicated reasoning component actually outperform a homogeneous baseline on logic problems — specifically problems that require generalizing to entities never seen during training?

3.1 Setup

We used GPT-2 (117M parameters, Radford et al., 2019) as a frozen language backbone — meaning its weights never changed during any of our training. GPT-2 converts the text of each puzzle into a numerical representation. Both the baseline and the reasoning component then operate on that same fixed representation. GPT-2 is the shared input layer for both; what differs is entirely what comes after it.

- Baseline: a standard 3-layer MLP (214K trainable parameters) trained on mean-pooled GPT-2 hidden states. An MLP — Multi-Layer Perceptron — is the simplest neural network design: layers of neurons fully connected to the next layer, no memory, no iteration, no special structure. It represents the default: what you get with no architectural ideas.
- Reasoning Component: a dedicated module (272K trainable parameters) with ternary weights, explicit memory cells, iterative reasoning steps, and a confidence-gated early exit.

The task was multi-step logical inference across seven reasoning types: syllogisms, transitive chains up to four steps, contrapositives, negation, and mixed reasoning. Training used one fixed set of entity names. Test data used entirely different entity names — never seen during training. This design tests whether the model learned logical structure or surface patterns.

Full code is available at: github.com/orbits64/project-synapse

3.2 Reasoning Component Design

The reasoning component takes the mean-pooled GPT-2 output as input and processes it through N iterative steps. At each step it reads from and writes to explicit memory cells using ternary-weight addressing. A GRU-style update integrates the memory readout with the current hidden state. A confidence gate allows early exit on easier problems — harder problems automatically receive more computation.

Ternary weights use a straight-through estimator during training to allow gradient flow while maintaining the discrete constraint at inference. Quantization threshold was 0.1. After training, weights converged to a stable 41% / 17% / 42% distribution across {-1, 0, +1}. The 17% zero proportion represents learned sparsity — not a random artifact.

3.3 Results

Both runs used early stopping with patience 40. Results were consistent across runs and across different random seeds.

Table 1: Project Synapse Results

Run 1 (seed 42):	Baseline 75.1%	Reasoning Component 87.3%	+12.2%
Run 2 (seed 12):	Baseline 68.5%	Reasoning Component 88.1%	+19.6%

The reasoning component uses more steps on harder problems. On four-step transitive chains it consistently uses the maximum step count. On simpler syllogisms the confidence gate triggers early exit at step 2-3. This adaptive behavior — more computation for harder problems — is consistent with genuine multi-step inference rather than pattern matching.

The improvement holds specifically on entity names the model has never seen. Surface-form memorization is ruled out. The reasoning component learned the logical structure of the problems.

> *Across both runs, the reasoning component converged to a similar test accuracy (87.3% and 88.1%). The baseline varied more significantly (75.1% and 68.5%). The reasoning component is more stable across different random initializations. The baseline is more sensitive to how it starts. That robustness is itself evidence that the architectural structure is doing real work — not the specific values the weights happened to start at.*

3.4 What This Means

A structurally distinct reasoning component, at comparable parameter count, outperforms a homogeneous baseline on reasoning tasks. The reasoning component has 27% more trainable parameters than the MLP baseline, and the obvious question is whether that explains the gap. It does not. The MLP reached 100% training accuracy by epoch 60 — meaning it had fully saturated its capacity on the training data — and yet its test accuracy peaked at 75% and declined from there. The MLP was not held back by a lack of parameters. It was held back by a lack of structure. The reasoning component, with its iterative steps and memory, kept improving on test data past the point where the MLP had already given up.

An open question raised by these results: if the reasoning component generalizes better on novel logic problems, does it also generalize better on reasoning-heavy coding tasks — debugging, algorithm design, multi-step problem solving? That hypothesis is not tested here, but it is a natural direction. The current results do not prove it; they motivate testing it.

4. Proposed Design: Two Parallel Components

Motivated by the Synapse results, we describe the full design we are building toward. This is a proposal grounded in the experimental evidence above. The implementation in progress at Orbits Systems is called Gravitas.

4.1 Core Idea

Two components, running in parallel, each doing one thing well:

- Reasoning Component: ternary weights {-1, 0, +1}, iterative, memory-augmented. Handles logical inference, planning, and working memory. Dense where density matters.
- Language Component: float16, smaller. Handles encoding and decoding natural language. Does not reason — it translates between language and the representations the reasoning component works with.

The communication is bidirectional. The language component encodes input and feeds semantic context to the reasoning component. The reasoning component runs its iterative process and feeds decisions back to guide generation. Both run in parallel, synchronized at defined intervals.

This is not a Mixture of Experts design. MoE routes the same kind of computation to different sub-networks. This design uses fundamentally different weight representations for fundamentally different functions.

4.2 Memory

- Working Memory: the ternary memory cells in the reasoning component. Fast, volatile, cleared between tasks.
- Long-term Memory: compressed embedding vectors persisted across sessions. Written after task completion. Retrieved by content similarity at the start of new tasks.

4.3 Training Strategy

The two components are pre-trained independently, then fine-tuned together. The reasoning component trains on structured logic puzzles with verified answers and a controlled difficulty curriculum. The language component trains on high-quality text. Quality over quantity, following the principle demonstrated by Gunasekar et al. (2023).

An open challenge: joint fine-tuning from independently pre-trained components introduces gradient balancing problems not yet fully addressed. This is noted as a limitation in Section 5.

4.4 On Scale

If architectural specialization improves reasoning efficiency at small scale — as Synapse suggests — then scaling a specialized architecture should compound that advantage. A system with dedicated ternary reasoning components scaled to the parameter count of current large models would, under this hypothesis, outperform a homogeneous architecture of the same size

on reasoning benchmarks. This is a hypothesis, not a proven result. It is the direction Gravitas is designed to test.

5. Limitations

The Synapse experiment used a frozen pre-trained language backbone. The reasoning component did not have to learn language and reasoning simultaneously from scratch. The full design, where both components train independently then jointly, presents challenges not yet addressed experimentally — particularly around gradient balancing during joint fine-tuning.

The evaluation tasks are synthetic. Strong performance on controlled logic puzzles does not guarantee strong performance on naturalistic open-ended reasoning, real-world coding tasks, or other domains. That evaluation is needed and has not yet been done.

This research was conducted by one person, independently, without access to large-scale compute or institutional resources. The experiments are small by current field standards. They are reported because the controlled design allows clean interpretation — not because scale is unimportant. Scaling these experiments is the next step.

The code for Project Synapse is a single Python file using PyTorch and a frozen GPT-2 backbone. It is available openly. The results are reproducible from that file with a fixed seed.

6. Conclusion

A dedicated ternary-weight reasoning component outperforms a homogeneous baseline by 12.2% on novel multi-step logic problems at comparable parameter count. The improvement comes from specialization, not size.

The proposed design builds on this result: two parallel components, each with the weight structure suited to its function. Ternary weights for reasoning. Float16 for language. Each doing one thing well. Not one painting — two pieces of a puzzle that fit together.

AI has been one painting for long enough. It is time to look at the pieces.

Code: github.com/orbits64/project-synapse

References

Gunasekar, S., Zhang, Y., Aneja, J., et al. (2023). Textbooks are all you need. arXiv:2306.11644.

Li, F., Zhang, B., & Liu, B. (2016). Ternary weight networks. arXiv:1605.04711.

Radford, A., Wu, J., Child, R., et al. (2019). Language models are unsupervised multitask learners. OpenAI Blog.

Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. NeurIPS 35.

Zhu, C., Han, S., Mao, H., & Dally, W. J. (2016). Trained ternary quantization. arXiv:1612.01064.