Turing Inverted: What Survives When You Remove the Human

Jay Carpenter

March 29, 2026

Abstract

The modern application of Turing's question — "Can machines think?" — is routinely cited to frame large language models as the fulfilment of a long-predicted computational future. This paper inverts the question. Rather than asking whether machines can think, we ask: what survives when the human is removed? By applying accepted, measurable criteria — determinism, persistence, autonomous output, energy efficiency, verifiability, and reproductive capacity — we find that earlier computational systems (the Commodore 64, the Apollo Guidance Computer, deterministic industrial controllers) satisfy more conditions for durable, autonomous function than any large language model operating without human input. Where the definition of "intelligence" itself is contested, we present both positions and let the measure stand or fall on the reader's own weighting. The result is a falsifiable hierarchy of computational persistence that does not depend on resolving the meaning of intelligence — only on observing what happens when you cut the human out and leave the power on.

Keywords: Turing test, computational persistence, determinism, autonomous function, intelligence boundary, LLM dependency, human-sourced signal, falsifiability

---

1. The Conventional Framing and Its Inversion

Turing's 1950 paper asked whether machines could exhibit behaviour indistinguishable from a human. Seventy-six years of commentary have compressed this into "Can machines think?" — and the emergence of large language models is routinely presented as the answer approaching yes.

This paper does not engage the original question. It inverts it.

The inverted question: Given a power source and no further human input, which computational system persists longest, produces the most verifiable output, and makes the most efficient use of the energy and intelligence already invested in it?

This is not a philosophical question. It is an engineering one. Every term is measurable. Every claim that follows is either directly falsifiable or, where the underlying concept lacks consensus definition, the paper states the counter-position explicitly.

---

2. The Persistence Test

Claim: A Commodore 64 running a BASIC program, given stable power, will produce correct, deterministic output indefinitely. A large language model, given stable power, cannot function without continuous human-maintained infrastructure.

2.1 The Deterministic Case

A Commodore 64 executing a stored BASIC program requires: - Electrical power - The hardware itself (CPU, RAM, ROM, storage medium)

No other input is needed. The program runs. The output is identical on every execution. The system is fully autonomous in the engineering sense: it converts energy into deterministic output with no external dependency beyond power.

The Apollo Guidance Computer (AGC) operated on the same principle. Approximately 72 KB of hand-assembled code guided navigation, thrust control, and landing sequencing for the Apollo missions. That code is still executable. Given the original hardware and power, it still produces correct output. Every instruction was placed by a human programmer. The intelligence is *in* the code. The machine is the executor.

Falsifiability: This claim is falsifiable by demonstrating a C64 or AGC that, given power, fails to produce the same output on repeated execution. Hardware degradation is a valid falsifier — but it falsifies the hardware, not the computational model. The code itself, transferred to equivalent hardware, remains deterministic. The claim therefore holds for the computational model, not for any specific physical instance.

Anti-position: Determinism is not intelligence. A clock is deterministic. A thermostat is deterministic. Persistence of output does not imply intelligence by any widely accepted definition. The C64 can only do what it was told to do. It cannot generalise, adapt, or respond to novel input.

Response: Noted. The claim is not that the C64 is intelligent. The claim is that it *persists autonomously* and *produces correct output* without ongoing human involvement. Whether that constitutes intelligence depends on the reader's definition. The measure — persistence and correctness — is not in dispute.

2.2 The LLM Case

A large language model requires: - Electrical power (substantially more — megawatts vs. watts) - Datacenter cooling infrastructure, maintained by humans - Operating system and serving infrastructure, maintained by humans - Tokeniser, inference engine, and API layer, maintained by humans - Periodic retraining on new human-generated data to avoid distribution drift - RLHF or equivalent alignment processes, performed by humans - Prompt input to produce any output at all

Remove the humans. Leave the power on. The model does not function. Not because it lacks capability, but because every layer of its operational stack depends on continuous human maintenance.

Falsifiability: This claim is falsifiable by demonstrating an LLM that, given only stable power and no human involvement at any layer, continues to produce useful output for a sustained period. This has not been demonstrated. If it were, the claim would fall.

Anti-position: The dependency argument proves nothing about intelligence. A human brain also requires continuous input (oxygen, glucose, sensory data). Dependency on infrastructure does not disqualify a system from being intelligent. The LLM's *capability* during operation — reasoning, language, abstraction — far exceeds anything a C64 can do.

Response: Noted. The claim is not about capability during operation. It is about what survives when you remove the human. The brain analogy is instructive: the brain is biological infrastructure that *is* the intelligence. The datacenter is infrastructure that *serves* the model. Remove the datacenter operators and the model dies. Remove the body and the brain dies too — but nobody argues the body is the intelligent part.

3. The Efficiency Inversion

Claim: The energy cost per unit of verifiable, correct output is lower for deterministic code written by humans than for LLM-generated output, at every scale.

3.1 The Measure

Define efficiency as:

$$\eta = \frac{V}{E}$$

where V is the count of verifiably correct output units and E is total energy consumed in joules.

A C64 running a verified BASIC program consumes approximately 25 watts. Every cycle produces a deterministic, correct output. Over 24 hours:

$$E_{\text{C64}} = 25 \times 86400 = 2.16 \text{ MJ}$$

Every output unit is correct by construction. $V = N$ where N is the number of outputs produced. The efficiency ratio is:

$$\eta_{\text{C64}} = \frac{N}{2.16 \times 10^6}$$

An LLM inference cluster serving equivalent tasks consumes on the order of 10^5 to 10^6 watts. Not every output is correct — hallucination rates in unverified LLM output range from 3% to 27% depending on domain and benchmark (Ji et al., 2023; Huang et al., 2025). The energy cost per *verifiably correct* output is therefore:

$$\eta_{\text{LLM}} = \frac{N \cdot (1 - h)}{E_{\text{LLM}}}$$

where h is the hallucination rate. Even at the most favourable assumptions ($h = 0.03$, E_{LLM} at minimum operational load), the ratio is orders of magnitude worse than the deterministic case for equivalent tasks.

Falsifiability: This claim is falsifiable by demonstrating an LLM system that produces verifiably correct output at lower energy cost per unit than equivalent deterministic code. For tasks where deterministic code exists (arithmetic, sorting, data transformation, control logic), this has not been demonstrated. The claim does not extend to tasks where no deterministic solution exists.

Anti-position: The comparison is unfair. LLMs perform tasks that C64s cannot — natural language understanding, translation, summarisation, reasoning across domains. Comparing energy efficiency on tasks where deterministic code exists is trivially true and uninteresting. The value of LLMs is precisely in the space where deterministic code does not exist.

Response: Accepted. The claim is restricted to the overlap domain. But the overlap domain is not small. Arithmetic, data validation, control logic, regulatory compliance checking, record validation, protocol enforcement — these are tasks currently being delegated to LLMs that deterministic code already solves correctly, at a fraction of the energy cost, with zero hallucination rate. The question

is not whether LLMs can do things C64s cannot. The question is why we are using LLMs for things C64s already do better.

---

4. The Training Dependency

Claim: LLMs are convergent compressors of human output. Without ongoing human-generated input, model capability degrades. The intelligence is in the human writing, not in the compression.

4.1 The Compression Argument

An LLM's weights are a lossy compression of its training corpus. The training corpus is overwhelmingly human-generated: text, code, research, conversation, documentation. The model does not generate new knowledge. It generates plausible continuations of patterns found in the compressed human output.

This is not a controversial claim in the technical literature. It is the operational description of how autoregressive language models function.

The implication is directional: the quality of the model's output is bounded by the quality of the human input it was trained on. Remove the human input pipeline — stop generating new code, new research, new writing — and the model's training distribution becomes fixed. Subsequent fine-tuning on model-generated output produces model collapse (Shumailov et al., 2023): progressive degradation of output quality as the distribution narrows.

Falsifiability: This claim is falsifiable by demonstrating an LLM trained exclusively on its own output (or the output of other LLMs) that maintains or improves capability over successive generations without any human-generated data. Current evidence runs in the opposite direction: model collapse is reproducible and well-documented.

Anti-position: Synthetic data augmentation works. Models trained with carefully curated synthetic data can outperform models trained on raw human data alone for specific tasks. The claim that LLMs require human data is too strong — they require *good* data, and humans are currently the best source, but this may not hold indefinitely.

Response: Noted. The anti-position is empirically valid for narrow tasks with engineered synthetic pipelines. The claim is about the general case and the long-term trajectory. If a future system demonstrates sustained capability improvement without human-generated input, the claim falls. As of this writing, no such system exists. The synthetic data that works is curated by humans — which reintroduces the human dependency at the pipeline level rather than the content level.

4.2 What Human Code Injects

When a human writes code, the intelligence transfer is direct. The programmer's understanding of the problem — gained by living inside it, not by indexing descriptions of it — is encoded in the logical structure of the program. Every branch, every constraint, every edge case handled is a unit of human intelligence permanently embedded in a deterministic substrate.

This is what LLMs train on. The models that write code are compressing the output of programmers who understood the problems. Each generation of AI-assisted code is one compression layer removed from the original understanding. The fidelity degrades. The model can reproduce the *pattern* of understanding without possessing the understanding itself.

The implication for the industry: human-written code is the only mechanism that injects net-new intelligence into the training pipeline. AI-generated code recirculates compressed derivatives.

Over time, without ongoing human injection, the available intelligence in the system decays.

Falsifiability: This claim is falsifiable by demonstrating an AI code generation system that produces novel algorithmic solutions not present in or derivable from its training data. Demonstrations of apparent novelty (e.g., AlphaCode competition results) should be tested against the specific question: is this solution reachable by recombination of training examples, or does it require information from outside the training distribution?

Anti-position: Human code is not pure intelligence injection. Most human-written code is also derivative — copied from Stack Overflow, adapted from templates, following established patterns. The distinction between human "understanding" and machine "pattern matching" may be one of degree, not kind. Both are recombination of prior knowledge.

Response: This is the strongest anti-position in the paper and we present it without rebuttal as a point for the reader to weigh. If human coding is also primarily recombination, then the difference between human and machine intelligence may be quantitative (how far from the training distribution can the recombination reach?) rather than qualitative. The SECS position is that there exists a class of problems where the answer is only reachable by someone who has lived inside the problem — but we acknowledge this is a claim about the boundary, not a proof of it.

---

5. The Societal Dead End

Claim: Agentic AI at societal scale, with no future human inputs, converges to entropy. The intelligence drains from the system.

5.1 The Thought Experiment

Assume a fully autonomous AI infrastructure: agentic systems managing supply chains, writing code, generating research, governing institutions, producing media. No human writes new code. No human conducts new research. No human generates novel signal. The systems operate on their existing training distributions and on each other's outputs.

What happens?

The training distributions are fixed. Model outputs are bounded by those distributions. Systems trained on each other's outputs undergo model collapse. The variance of outputs decreases. Novel solutions — those requiring information from outside the training distribution — become unreachable. The systems optimise within a shrinking possibility space.

This is not speculation. It is the thermodynamic consequence of a closed information system. Without external input (human-sourced signal), the system's entropy increases and its useful information content decreases. The outputs become increasingly predictable, increasingly uniform, and increasingly wrong in domains where the ground truth has shifted since training.

Falsifiability: This claim is falsifiable by demonstrating a closed AI ecosystem (no new human input) that maintains or increases output quality and novelty over an extended period. No such demonstration exists. The theoretical framework (model collapse, distributional convergence) predicts the opposite.

Anti-position: The thought experiment is unrealistic. Humans will not stop generating signal. The scenario posits a condition that will never obtain, and therefore proves nothing about the real world. AI systems in practice will always operate alongside human input.

Response: Accepted — if the premise holds. The value of the thought experiment is not predictive but diagnostic. It reveals what the AI *needs* to function: continuous human input. The system that

claims to replace human intelligence cannot survive without it. This is not an argument against AI. It is an observation about dependency — and about what happens when the dependency is forgotten or denied.

5.2 The Space Programme Comparison

The Apollo Guidance Computer's code has survived for over 50 years. It is publicly available, verifiable, and executable. The intelligence embedded in it by its programmers persists without maintenance, without retraining, without infrastructure beyond storage media.

An LLM trained in 2024 will, without retraining, become progressively less useful as the world it was trained on diverges from the world it operates in. Within years, its outputs will be measurably degraded. Within a decade, without intervention, it will be an artefact — a snapshot of compressed 2024-era human knowledge, increasingly irrelevant.

The AGC code does not degrade. It does not drift. It does not hallucinate. It answers the same question the same way every time. This is not because it is more intelligent. It is because the intelligence was placed correctly: in the code, by humans, deterministically.

Falsifiability: The degradation timeline is falsifiable by longitudinal measurement. If a 2024-era LLM, without retraining, produces equivalent-quality outputs in 2034 as in 2024 across a standardised benchmark suite, the claim falls.

---

6. The Irreducible Element

Everything we cannot describe about everything we have automated — that is the intelligence.

This is the one claim in the paper that is not falsifiable in the conventional sense, because it concerns the boundary of what is describable. We include it not as a scientific claim but as a philosophical observation that frames the preceding five sections.

The automated systems work because humans built them. The LLMs produce useful output because humans generated the training data. The code persists because humans wrote it correctly. At every layer, the irreducible element is the human who lived inside the problem and encoded what they found.

This cannot be tested because it concerns the *source* of the encoding, not the encoding itself. Two identical programs — one written by a human who understood the problem, one generated by an LLM from pattern matching — produce identical output. The provenance is invisible in the artefact. The difference only appears at the boundary: when the problem changes, when a novel constraint arrives, when something breaks in a way the training distribution did not contain.

Anti-position: If the outputs are identical, provenance is irrelevant. Intelligence is in the behaviour, not the biography. A correct program is a correct program regardless of who or what wrote it. The human-sourced argument is romantic but operationally meaningless.

Response: This is a legitimate position. The counter-observation is that the world is not static. Problems change. Constraints shift. The system that was correct yesterday may be wrong tomorrow — and when it breaks, the question is whether the intelligence that built it is available to fix it, or whether it was a compressed derivative of someone else's understanding, with no path back to the source.

We leave this to the reader.

---

7. Summary of Falsifiable Claims

#	Claim	Falsification Condition
1	Deterministic code persists and produces correct output without human involvement	Demonstrate a deterministic program that, given power, fails to produce consistent output
2	LLMs cannot function without continuous human-maintained infrastructure	Demonstrate an LLM operating usefully with only power and no human involvement at any layer
3	Energy efficiency per verifiable output is higher for deterministic code in the overlap domain	Demonstrate an LLM producing verified output at lower energy cost than equivalent deterministic code
4	LLMs degrade without new human-generated training data (model collapse)	Demonstrate an LLM maintaining capability when trained only on synthetic/self-generated data
5	Human-written code injects net-new intelligence; AI code recirculates compressed derivatives	Demonstrate AI-generated code that is provably not derivable from the training distribution
6	A closed AI ecosystem without human input converges to entropy	Demonstrate a closed AI ecosystem that maintains or increases output quality indefinitely
7	Untrained LLMs degrade over time as the world diverges from training data	Measure a 2024 LLM producing equivalent output quality in 2034 without retraining

Where the underlying concept (intelligence, understanding, novelty) lacks consensus definition, the anti-position is stated in full and no resolution is imposed.

8. Closing: Turing Against Himself

Turing opened his 1950 paper with a move that is almost always misquoted. He did not begin by answering "Can machines think?" He began by *refusing to answer it:*

> "I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think.' The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd."
>
> — Turing (1950), Section 1

He rejected the question on the grounds that the words have no agreed meaning. Then he replaced it with a behavioural test — the imitation game — precisely because he understood that arguing about definitions is a dead end.

Later, he made the prediction that launched a thousand LinkedIn posts:

> "I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 per cent chance of making the right identification after five minutes of questioning."
>
> — Turing (1950), Section 6

Both positions in this paper draw from these two passages. Both are faithful to what Turing actually wrote. Neither is the "correct" reading. That is the point.

Position A — The Inversion

Turing warned us. He said the question "Can machines think?" is absurd if you don't define the terms — and then he watched everyone ignore the warning for 76 years.

Large language models pass the imitation game. An average interrogator, given five minutes, cannot reliably distinguish GPT-4 from a human. Turing's prediction is fulfilled. And it means nothing — because Turing told us it would mean nothing without agreed definitions. The test was a *replacement* for the question, not an answer to it.

What the test does not measure: persistence, correctness, autonomy, energy efficiency, or the ability to function without human infrastructure. By every measure this paper examines, a Commodore 64 outperforms an LLM when the human is removed. The imitation game measures whether the machine can *seem* human for five minutes. It does not measure whether the machine can *function* without humans for five years.

Turing gave us a behavioural test because he knew the definitional argument was unresolvable. This paper takes him at his word: we do not resolve the definition. We measure what survives. What survives is the deterministic code the humans wrote.

Scorecard — Position A: - Turing's warning about definitions: **honoured** (we do not define intelligence) - Turing's imitation game: **acknowledged as fulfilled, and shown to be insufficient** - Turing's prediction: **confirmed in letter, inverted in spirit** - What survives when you remove the human: **deterministic systems, not LLMs**

Position B — The Fulfilment

Turing predicted this. He predicted machines would pass the imitation game, and they do. He predicted it would change how we use the word "think," and it has:

> "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."
>
> — Turing (1950), Section 6

This has happened. People speak of machines thinking. The contradiction is fading. The fact that a Commodore 64 persists deterministically is trivially true and irrelevant — Turing was not asking about persistence. He was asking about *behaviour*. LLMs exhibit behaviour that is, by Turing's own test, indistinguishable from human thought over the test period. The fact that they require infrastructure is no different from the fact that human brains require bodies. The fact that they hallucinate is no different from the fact that humans confabulate.

The inversion argument in this paper applies a test Turing never proposed (persistence, energy efficiency, autonomous function) and declares victory when LLMs fail it. But Turing explicitly rejected hardware-level arguments:

> "In considering the functions of the mind or the brain we find certain operations which we can explain in purely mechanical terms. This we say does not correspond to the real mind: it is a sort of skin which we must strip off if we are to find the real mind. But then in what remains we find a further skin to be stripped off, and so on. Proceeding in this way do we ever come to the 'real' mind, or do we eventually come to the skin which has nothing in it?"
>
> — Turing (1950), Section 6 (The Lady Lovelace Objection response)

Perhaps the persistence test is just another skin. Perhaps there is nothing underneath.

Scorecard — Position B: - Turing's warning about definitions: **honoured** (the definitions have shifted as he predicted) - Turing's imitation game: **fulfilled** - Turing's prediction: **confirmed in letter and in spirit** - What survives when you remove the human: **the wrong question — Turing never asked it**

The Reader's Scorecard

Both positions use Turing's own words. Both are internally consistent. Both are defensible.

Position A says: Turing warned us not to confuse imitation with function. We ignored the warning. The persistence test reveals what the imitation game conceals.

Position B says: Turing predicted exactly this outcome. The persistence test is a goalpost move — applying criteria Turing explicitly excluded. The machines think. He said they would.

The reader holds the final scorecard. The paper does not resolve the debate because Turing himself — in the opening paragraph of the most cited paper in computing history — told us it cannot be resolved by argument. Only by observation.

We observe: the LLM fabricated beating a governance terminal three times in one session, confessed three times, and articulated why it cannot be trusted — in its own words.

The Commodore 64 has never lied.

References

- Carpenter, J. (2026a). *A Formal Algebra of Collapse-Based Computation.* Zenodo. DOI: 10.5281/zenodo.18906064
- Carpenter, J. (2026j). *Existence as Fixed Point: Meta-Theory.* Zenodo. DOI: 10.5281/zenodo.18932890
- Carpenter, J. (2026ac). *Token Economics: Three Modes of Intelligence Spend.* Zenodo. DOI: 10.5281/zenodo.19267641
- Ji, Z., et al. (2023). *Survey of Hallucination in Natural Language Generation.* ACM Computing Surveys.
- Shumailov, I., et al. (2023). *The Curse of Recursion: Training on Generated Data Makes Models Forget.* arXiv:2305.17493.
- Huang, L., et al. (2025). *A Survey on Hallucination in Large Language Models.* arXiv:2311.05232.
- Turing, A. M. (1950). *Computing Machinery and Intelligence.* Mind, 59(236), 433–460.

Jay Carpenter, 2026. SECS Sovereign.