

Semantic Field Execution:

A Substrate for Field-Native, Transformer-Decoupled Inference

Ryan Shamim¹

¹Anima Core Inc., ryan@animacore.ai

December 2025

Abstract

Most work on inference efficiency assumes a fixed execution primitive, namely runtime evaluation of large transformer models. Under this assumption, reducing the cost of inference often increases aggregate compute demand, a phenomenon commonly described as an inference-efficiency paradox. In this paper, we examine a different regime. We introduce Semantic Field Execution (SFE), an execution substrate in which transformers are used only offline for semantic sculpting, while all runtime inference is performed via field-native inference on a compact semantic field using shallow, bounded operations. We define a corresponding Semantic Field Runtime (SFR) and describe ANI as a concrete implementation. We argue that SFE constitutes a substrate shift rather than an optimization, clarify how it violates the assumptions underlying transformer-specific efficiency paradoxes, and articulate explicit, operational falsifiability conditions that bound its applicability. Our goal is not to claim universal superiority, but to precisely define a new execution regime for inference and establish a framework for its evaluation.

1 Introduction

Transformer architectures currently serve as the dominant substrate for machine learning inference [1]. Their success across language, vision, and multimodal tasks has driven widespread deployment, accompanied by significant computational cost. Scaling laws have further demonstrated that model performance improves predictably with increased compute [2, 3]. In response, a large body of work has focused on optimizing transformer inference through quantization [4, 5], pruning, kernel fusion, speculative decoding, caching, and architectural refinements [6].

Despite these advances, aggregate compute usage has continued to rise. Cheaper inference has enabled denser deployment, higher invocation frequency, and new application classes. This dynamic is often framed as an *inference-efficiency paradox*, echoing rebound effects observed in other technological systems.

Most discussions of this paradox implicitly assume that inference remains bound to transformer execution. In this paper, we examine a regime in which that assumption no longer holds.

We introduce **Semantic Field Execution (SFE)**, an execution substrate in which the transformer is removed from the runtime inference loop. Instead, a high-capacity transformer is used offline to extract and compress task-relevant semantic structure into a compact *semantic field*. Runtime inference then operates exclusively on this field via *field-native inference*, without executing the transformer itself.

1.1 Contributions

This paper makes three contributions:

1. It defines Semantic Field Execution as a distinct inference substrate.
2. It introduces the Semantic Field Runtime as an execution model for field-native inference, with ANI as a concrete implementation.
3. It reframes inference-efficiency paradoxes by explicitly identifying the assumptions they depend on and showing how SFE violates them.

Our aim is conceptual precision rather than empirical finality.

2 Background and Motivation

2.1 Transformer Inference as a Fixed Runtime Primitive

In current systems, transformers function simultaneously as representational learners and runtime execution engines. Inference consists of token-sequential evaluation of a deep parameterized function dominated by matrix multiplication and attention. Performance is measured relative to the cost of executing this same primitive.

Optimization efforts preserve transformer execution while reducing its cost, implicitly treating the execution substrate as fixed.

2.2 Efficiency and Rebound Effects

Empirically, reductions in inference cost have expanded deployment rather than constrained it. Lower marginal costs enable more users, more agents, and finer-grained applications. This pattern mirrors historical dynamics in computing, where efficiency improvements frequently increase total consumption [7, 8].

Crucially, these analyses presume continuity of the runtime substrate.

2.3 Questioning the Substrate Assumption

If inference no longer consists of executing a transformer, the object being optimized changes. Cost curves, latency constraints, hardware mappings, and economic dynamics must be reconsidered. Semantic Field Execution explores this possibility by separating semantic learning from runtime execution.

3 Semantic Field Execution

3.1 Definition

Semantic Field Execution (SFE) is an inference substrate in which runtime execution operates on a compact *semantic field* rather than on a large neural network. A semantic field is a structured, low-dimensional representation encoding task-relevant meaning extracted from a high-capacity model.

Concretely, a semantic field may take the form of a fixed-dimensional vector, tensor, or structured state derived from intermediate activations of a transformer and compressed for deployment. The field is designed to be sufficiently expressive for a target task class while remaining compact and stable.

In SFE, transformers are used only offline. They are not executed during runtime inference.

3.2 Semantic Sculpting

We refer to the offline process of extracting and compressing task-relevant structure from transformer activations as *semantic sculpting*. By sculpting, we mean identifying, filtering, and compressing activation patterns that encode task-aligned meaning into a deployable representation.

This process is unconstrained by runtime latency or cost and may involve training sweeps, probing, or analysis procedures. Its output is a semantic field suitable for field-native inference.

3.3 Semantic Field Runtime

A **Semantic Field Runtime (SFR)** defines how field-native inference is executed. It specifies:

- the representation of semantic fields,
- permissible lookup and update operations,
- constraints on computational depth, state mutation, and latency.

An SFR is analogous to a virtual machine [9] or GPU runtime [10]. It defines an execution model independent of any specific hardware instantiation.

3.4 AN1 as an Implementation

AN1 is a concrete implementation of an SFR. It supports semantic sculpting, field formation, and runtime inference via bounded field operations. AN1 is an implementation, not the substrate itself.

4 Pipeline Overview

The SFE pipeline consists of three stages:

- 1. Offline Semantic Sculpting.** Raw activations are extracted from a transformer during training or inference sweeps. These activations encode task-relevant semantic structure.
- 2. Semantic Field Formation.** The raw semantic field is compressed into a compact representation suitable for deployment. Compression may involve dimensionality reduction, quantization, or learned encoding.
- 3. Field-Native Inference.** Runtime inference consists of shallow, bounded operations over the compressed semantic field, such as lookups, interpolations, or shallow compositions. No transformer execution occurs at runtime.

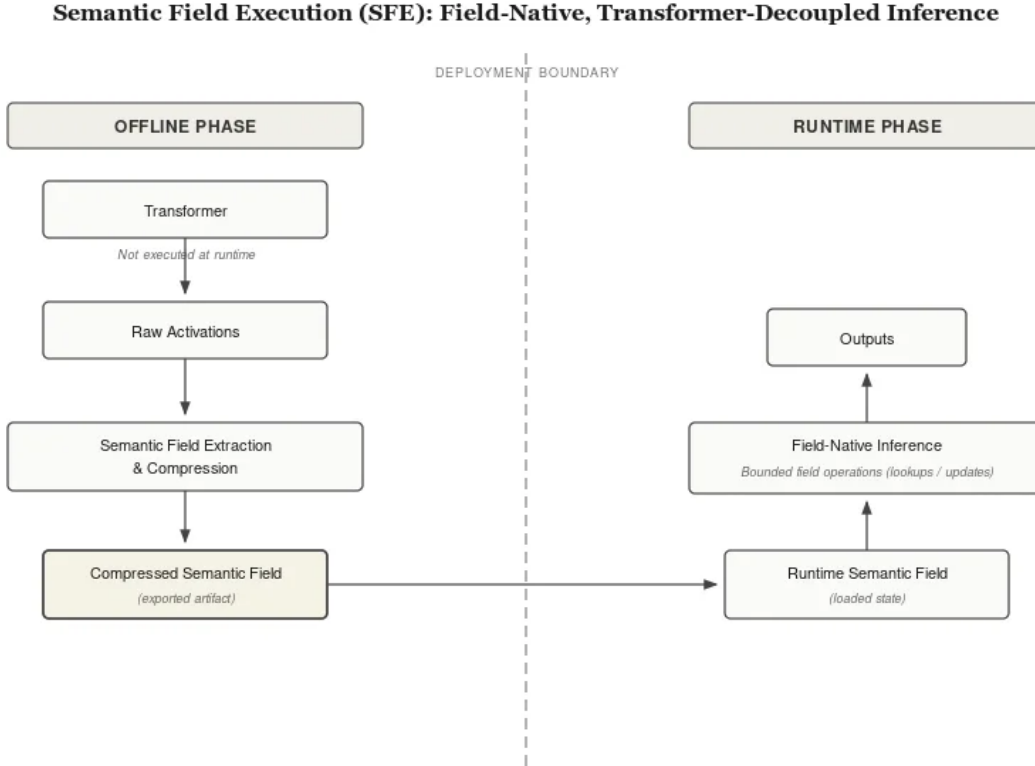


Figure 1: Semantic Field Execution (SFE): Field-Native, Transformer-Decoupled Inference. The transformer is used only offline for semantic sculpting. Runtime inference operates exclusively on the compressed semantic field via bounded field operations.

5 Substrate Shift Rather Than Optimization

SFE represents a change in the execution primitive, not an optimization of an existing one. Inference optimizations preserve transformer execution and seek to reduce its cost. SFE replaces transformer execution with field-native inference.

This distinction mirrors historical transitions such as [11]:

- *Interpretation to compilation*, which enabled orders-of-magnitude speedups for repeated execution,
- *CPU execution to GPU execution* [10], which enabled massive parallelism for data-parallel workloads,
- *Scalar to vectorized processing*, which exploited regular computation patterns via SIMD.

SFE similarly exploits a structural property: task-relevant meaning often concentrates in intermediate representations that can be extracted and executed upon directly.

6 Reconsidering Inference-Efficiency Paradoxes

6.1 Transformer-Specific Paradox

The standard inference-efficiency paradox presumes that inference consists of executing a transformer. Under SFE, this assumption is violated. Cheaper field-native inference does not increase transformer execution because transformers are not part of the runtime loop.

In this narrow, transformer-specific sense, the paradox does not apply.

6.2 Generalized Rebound and Compute Migration

A broader rebound effect may still occur. If field-native inference dramatically lowers the cost of useful cognition, demand for cognitive services may expand. Compute may migrate toward data preparation, semantic sculpting, field updates, and broader application deployment.

SFE therefore reframes, rather than eliminates, economic questions about efficiency and demand. The relevant commodity shifts from transformer inference to field-based cognitive execution.

7 Falsifiability and Boundary Conditions

SFE should be rejected as a substrate if:

- semantic fields cannot remain compact, for example requiring more than a small fraction of the original model parameters,
- maintaining performance requires unbounded depth or cost in runtime field operations, for example exceeding a fixed number of sequential operations or latency thresholds,
- transformers must be reintroduced into the runtime loop,
- or the approach fails to generalize beyond narrowly scoped domains.

Open-ended reasoning, rapidly shifting domains, or certain multimodal tasks may resist stable semantic field representations. SFE is therefore best understood as a regime whose applicability varies across tasks rather than a universal replacement for transformer inference.

8 Implications

Semantic Field Execution has implications across multiple layers of the stack.

Hardware. SFE motivates specialized accelerators for field-native operations, such as Field Processing Units optimized for lookup, interpolation, and shallow composition.

Software. New tooling is required for debugging, profiling, and versioning semantic fields, analogous to how GPU computing required new software abstractions.

Economics. Serving cost structures shift from transformer inference to field lifecycle management, altering incentives around deployment, updates, and scaling.

9 Related Work

SFE builds on insights from representation probing [12, 13], model distillation [14, 15], frozen-feature classifiers [16], and ML systems research. Prior work demonstrates that task-relevant information is often concentrated in intermediate activations and can be exploited by shallow models. Recent mechanistic interpretability work has further revealed that transformer internal representations encode structured, recoverable information [17], and that feed-forward layers function as key-value memories storing factual associations [18]. However, these approaches typically treat representations as auxiliary artifacts rather than as the primary runtime execution substrate.

Distillation approaches still execute a neural model at runtime, albeit a smaller one. In contrast, SFE executes field-native operations on a semantic representation, constituting a different computational primitive.

Classical work on semantic networks and knowledge representation shares philosophical affinities but differs operationally. SFE defines a concrete execution model for learned semantic fields rather than symbolic graphs.

10 Conclusion

We have introduced Semantic Field Execution as a substrate for field-native, transformer-decoupled inference. By separating offline semantic sculpting from runtime execution, SFE challenges the assumption that inference must execute large neural networks. This shift clarifies the scope of transformer-specific efficiency paradoxes and reframes broader economic questions about AI deployment.

AN1 demonstrates that this regime is implementable. Whether SFE becomes a dominant substrate will depend on empirical validation, system integration, and economic dynamics. This paper establishes the vocabulary, boundaries, and evaluative criteria for that exploration.

References

- [1] Vaswani, A., Shazeer, N., Parmar, N., et al. Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [2] Kaplan, J., McCandlish, S., Henighan, T., et al. Scaling Laws for Neural Language Models. *arXiv:2001.08361*, 2020.
- [3] Hoffmann, J., Borgeaud, S., Mensch, A., et al. Training Compute-Optimal Large Language Models. *arXiv:2203.15556*, 2022.
- [4] Dettmers, T., Lewis, M., Belkada, Y., et al. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *NeurIPS*, 2022.
- [5] Frantar, E., Stock, P., Alistarh, D. GPTQ: Accurate Post-Training Quantization for Generative Pretrained Transformers. *ICLR*, 2023.
- [6] Weng, L. Inference Optimization Techniques for Large Language Models. *lilianweng.github.io*, 2023.
- [7] Alcott, B. Jevons' Paradox. *Ecological Economics*, 2005.
- [8] Koomey, J., Berard, S., Sanchez, M., Wong, H. Implications of Historical Trends in the Electrical Efficiency of Computing. *IEEE Annals of the History of Computing*, 2011.
- [9] Rosenblum, M., Garfinkel, T. Virtual Machine Monitors: Current Technology and Future Trends. *IEEE Computer*, 2005.
- [10] Owens, J. D., Houston, M., Luebke, D., et al. GPU Computing. *Proceedings of the IEEE*, 2008.
- [11] Patterson, D. A., Hennessy, J. L. Computer Architecture: A Quantitative Approach. *Morgan Kaufmann*, 2017.
- [12] Alain, G., Bengio, Y. Understanding Intermediate Layers Using Linear Classifier Probes. *ICLR Workshop*, 2017.
- [13] Hewitt, J., Manning, C. D. A Structural Probe for Finding Syntax in Word Representations. *NAACL*, 2019.
- [14] Hinton, G., Vinyals, O., Dean, J. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, 2015.
- [15] Sanh, V., Debut, L., Chaumond, J., Wolf, T. DistilBERT: A Distilled Version of BERT. *NeurIPS Workshop*, 2019.
- [16] Chen, T., Kornblith, S., Norouzi, M., Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. *ICML*, 2020.
- [17] Elhage, N., Nanda, N., Olsson, C., et al. A Mechanistic Interpretability Analysis of Grokking. *arXiv:2301.05217*, 2023.
- [18] Geva, M., Schuster, R., Berant, J., Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. *EMNLP*, 2021.