# The Function Model: Human-Like Learning Through Streaming Functional Updates

John Harby

November 2025

Abstract

Traditional machine learning systems rely on gradient-based optimization, batch training, and high computational cost. The Function Model represents a fundamentally different paradigm, where learning is expressed as localized functional updates, applied instantly and deterministically. This enables continuous adaptation, streaming training, drift-free behavior, and human-like incremental refinement. This paper provides a conceptual overview of the architecture, using simple examples to illustrate patch behavior, inference structure, and stability properties. Implementation details are omitted and available under NDA for organizations evaluating deployment.

## Introduction

Contemporary machine learning depends on iterative training, repeated passes through large datasets, and gradient descent optimization. Learning occurs in periodic cycles, often requiring significant GPU resources. Inference and training exist as separate operational modes.

The Function Model introduces a different approach. Instead of adjusting a high-dimensional parameter space, the model learns by applying direct functional modifications of the form

$$f(x) := y,$$

localized to exactly the input-output pair being taught. This resembles human one-shot learning, where a correction updates a fact or skill rather than triggering global re-optimization. A simple example being, “I look at my table and immediately learn what is on it. I don’t need to wait for batch updates for this”.

## Local Patches and Global Stability

Consider the function

$$f(x) = \sin(x).$$

Suppose a new training signal indicates that the value at $x = 2$ should be 3. Traditional ML incorporates this example into a dataset, computes gradients, and updates parameters throughout the model. This global diffusion produces instability and drift.

A function model system instead records a local functional modification:

$$f(x) = \begin{cases} \sin(x), & x \neq 2, \\ 3, & x = 2. \end{cases}$$

The modification is localized, deterministic, reversible, auditable, and has no effect on unrelated regions of the function space. Over time the model accumulates a structured set of modifications representing precise corrections.

$$= \{g(x) \; else \; \sin(x), x \notin S, f(x)$$

Here S is the set of all prior training domain values while g(x) will return the training assigned for each x in that domain. If training (time1, 2, 4), (time2, 2, 6) and (time2, 3, 5) where time2 $>$ time1 have occurred then $S = \{2, 3\}$ and $g(2) = 6$ while $g(3) = 5$. Of course prior values of everything concerned are all logged.

# Streaming Learning

Under the Function Model, learning is immediate. A new training pair $(x, y)$ is incorporated through a localized update, without batching or iterative optimization. This produces several outcomes:

- Immediate adaptation: the model incorporates new information at once.
- No separation of training and inference: learning occurs continuously.
- No catastrophic forgetting: patches persist unless intentionally removed.
- Minimal compute: updates are effectively $O(1)$.

This architecture mirrors human cognition more closely than traditional ML: continuous refinement without global retraining.

# Inference in a Function Model System

Function model inference proceeds through two conceptual layers:

- a base function $f_0(x)$ representing global structure,
- a patch layer $g(x)$ defined only on a finite domain $S$.

Inference evaluates

$$f(x) = \begin{cases} g(x), & x \in S, \\ f_0(x), & x \notin S. \end{cases}$$

This structure ensures determinism, stability, replay capability, and version-controlled behavior. Because updates do not diffuse globally, the system avoids drift entirely.

## High-Level Concurrency Considerations

In high-throughput environments, inference and learning may occur simultaneously. A conceptually appropriate approach is to maintain a model serving inference alongside a model receiving updates, with safe reconciliation between them. This avoids contention and preserves deterministic behavior.

The mechanisms vary by implementation and are not described here. Details can be discussed under NDA after patent processes are complete.

## Implications for Agentic and Autonomous Systems

Drift is a critical failure mode in agent systems: memory drift, policy drift, and workflow desynchronization degrade reliability over time. The Function Model, with its localized and deterministic patching, provides drift-free continuous learning. This makes it well suited for:

- agentic operating systems,
- enterprise workflow automation,
- real-time adaptive decision systems,
- scientific and simulation environments requiring incremental correction.

## Conclusion

The Function Model replaces gradient descent with functional, patch-based learning. Updates are local rather than global, deterministic rather than stochastic, and streaming rather than batch oriented. This enables human-like learning behavior while ensuring long-term stability and auditability.

Organizations exploring implementation can request deeper architectural details under NDA. Patent pending.