Self-Organizing Mixture of Experts (SOME): A Framework for In-Inference Adaptation

Author: Focus Labs
Date: October 11, 2025

1. The Core Problems Addressed

A primary challenge in deploying large-scale neural networks is the tension between knowledge stability and adaptive plasticity. Three fundamental challenges motivate the SOME architecture:

- The VRAM Wall: Standard MoE models require all experts to be loaded into memory, making it prohibitively expensive to scale to millions of experts, as memory costs scale linearly with the number of experts.
- The Static Router & Catastrophic Forgetting: Traditional models have a static knowledge structure. Adapting them via fine-tuning often corrupts existing knowledge (catastrophic forgetting), while a lack of adaptation makes them brittle and unable to handle evolving data distributions.
- The Specialization Paradox: Existing MoE architectures suffer from two inefficiencies:
    - Knowledge Redundancy: Multiple experts waste parameters learning the same common-sense facts and basic grammar.
    - Knowledge Hybridity: Experts often fail to become truly specialized, remaining generalists and diluting the "expert" paradigm.

2. The Core Innovation: Decoupling Function from Address

The central thesis of SOME is that a model can achieve in-inference adaptation by modifying how it accesses its knowledge rather than changing the knowledge itself. This is accomplished by a strategic separation of an expert's immutable function from its mutable address.

- Static Function (The "What"): Each expert's core knowledge—its internal weights—is frozen after pre-training. This creates a stable and reliable library of specialized functions, directly solving the problem of catastrophic forgetting.
- Dynamic Address (The "Where"): Each expert is assigned a "routing key"—a small, low-dimensional vector representing its location in a high-dimensional conceptual space. These keys are plastic and are continuously updated during inference.

The model doesn't re-learn how to perform a task; it re-learns how to better organize its internal "address book" to find the right specialist for the task at hand.

3. Query Fall: The Paradigm of Knowledge Gravity

SOME introduces a fundamental shift in how routing is conceptualized. An incoming query does not get sent to an expert by a decision-making router; it falls toward the most relevant region of knowledge.

- Standard MoE (The Switchboard Operator): A typical router acts as a classifier, making a discrete decision (Query -> Decision -> Expert). This trained decision-maker is the source of the static routing problem.
- The SOME Paradigm (Knowledge Gravity): SOME removes the decision-maker. The query vector q and the expert keys k_i exist in the same semantic space. Routing is an

emergent property of proximity, found via a massive-scale similarity search. The relationship is Query -> Attraction -> Expert Cluster.

This shift to an emergent, gravity-based system unlocks true self-organization, sidesteps the static router problem, and enables greater scalability and efficiency through optimized search algorithms instead of intractable classification.

4. The Self-Organizing Mechanism: The Forces of Knowledge Gravity

The dynamic evolution of the key space is governed by a set of principled, gradient-free update rules that function as a simulated physical system.

- Attractive Forces (Consolidation):
  - Query Pull: An expert's key is pulled toward the centroid of the queries that activate it, ensuring experts drift toward the conceptual center of their specialization.
  - Peer Pull: Keys of co-activated experts are pulled closer together, forging "gravitational bonds" that create self-organizing "knowledge galaxies" of complementary skills.
- Stabilizing Forces (Equilibrium):
  - Usage Inertia ("Gravitational Mass"): The learning rates for key updates are scaled down by an expert's usage. Popular, generalist experts gain more "mass," anchoring the key space and preventing instability.
  - Repulsive Decay ("Dark Energy"): Keys of infrequently used experts are slowly decayed toward the origin. This prevents the gravitational collapse of all keys and serves as a pruning mechanism to recycle useless experts.

Related Work

- 2.1. Self-Organizing Systems: The "Query Pull" mechanism is a direct architectural integration of the learning rule from Learning Vector Quantization (LVQ) and Self-Organizing Maps (SOMs), augmented with the compound dynamics of Peer Pull, Inertia, and Decay.
- 2.2. Continual Learning (CL): SOME proposes an orthogonal approach to standard CL methods. By freezing expert weights and only updating addresses, it aims to mitigate catastrophic forgetting without requiring parameter expansion or replay buffers.
- 2.3. Mixture of Experts (MoE) Routing: SOME incorporates lessons from modern MoE load balancing. The "Usage Inertia" mechanism is a step in this direction, preventing expert "hotspotting." Our architecture is directly inspired by the scaling ambitions of models like Google Research's Mixture of a Million Experts (MoME), which highlighted the need to move beyond simple routing networks to handle a vast number of hyper-specialized experts. SOME provides a novel solution to MoME's core routing and infrastructure challenges.

Formal Definition of the SOME Architecture

1.0 Objective

To formally define a neural network architecture that separates an expert's static function from its dynamic address, enabling continuous, in-inference adaptation while maintaining a stable knowledge base.

2.0 Core Components

A SOME layer is a sub-module designed to replace the FFN block within a transformer architecture.

2.1 Static Components (The Knowledge Base)

These components are frozen after the initial pre-training phase.

- Shared Base FFN (FFN_base): A single, standard FFN exists per SOME layer. Its weights are frozen and shared by all experts in that layer.
- Expert Pool (E): A collection of M independent, parameter-efficient adaptation modules. $E = \{e_1, e_2, ..., e\_M\}$.
- Individual Expert (e_i): Each expert is a Low-Rank Adaptation (LoRA) module consisting of two small matrices, A_i and B_i. This parameter-efficient representation formally solves the VRAM Wall problem, allowing M to scale to millions.
    - Function: An expert e_i creates a specialized, residual transformation on the hidden state z. A possible function is:
      $e_i(z) = FFN\_base(z) + B_i(A_i z)$
    - State: The FFN_base and all LoRA matrices A_i and B_i are static and frozen post-training.

2.2 Dynamic Components (The Routing System)

These components are plastic and evolve during inference.

- Query Network (Q): A small neural network that maps an input hidden state x to a query vector q. Its function is to embed the input into the shared semantic space.
    - Function: $q = Q(x)$
    - State: The Query Network Q is frozen after the pre-training phase. This ensures the semantic space has stable coordinates, while the expert keys move within it.
- Key Store (K): A dynamically updatable key-value store containing the routing key k_i for each expert e_i, implemented using an efficient ANN index like FAISS.
    - $K = \{k_1, k_2, ..., k\_M\}$ where $k_i \in \mathbb{R}^{(d\_key)}$.
    - State: The keys k_i are plastic and evolve during inference.

3.0 Architectural Integration: The Forward Pass

1. Query Generation: $q = Q(x)$
2. Expert Scoring (ANN Search): Compute similarity scores $s_i = q \cdot k_i$ for all M experts, retrieving the TopK_indices (I) using an efficient ANN search.
3. Gating & Computation: Compute gating weights g_i via softmax on the scores of the selected experts. Process the input x in parallel through the selected experts: $y_i = e_i(x)$ for $i \in I$.
4. Output Combination: Compute the weighted sum $y\_SOME = \Sigma_{i \in I} g_i * y_i$.
5. Final Output: $y\_output = x + y\_SOME$ (Residual Connection).

4.0 Dynamic Update Mechanism: "Knowledge Gravity"

This process is gradient-free and occurs in a "synaptic consolidation" phase after a batch of data is processed.

- 4.1 Force 1: Query Pull (Relevance Attraction):
    - Update Rule: k_i_new = k_i_old + α * (q - k_i_old) for each (q, e_i) pair in the Top-K set.
- 4.2 Force 2: Peer Pull (Hebbian Co-activation):
    - Update Rule: k_i_new = k_i_old + β * (k_j_old - k_i_old) and k_j_new = k_j_old + β * (k_i_old - k_j_old) for each co-activated (e_i, e_j) pair.
- 4.3 The Formal Objective (Energy Function): The update rules can be understood as a gradient-free, online optimization of a global Energy Function E for the key space. The system's goal is to minimize this energy over time.
    - E = E_attraction + E_repulsion
    - Attraction Energy (E_attraction): A function of the distances between queries and their activated keys, and between co-activated peer keys. The Query and Peer Pull rules are practical steps to minimize this term.
    - Repulsion Energy (E_repulsion): A function that penalizes the system for lack of diversity by growing as the distance between any two keys shrinks. The Repulsive Force / Decay mechanism is a practical step to minimize this term, preventing systemic collapse.

5.0 Stability Mechanisms

These mechanisms ensure the long-term stability and equilibrium of the self-organizing key space.

- 5.1 Gravitational Mass (Usage Inertia): Modulates the learning rates based on expert usage, anchoring the system around frequently used, generalist experts.
    - Implementation: α_effective = α / (1 + usage(e_i)) and β_effective = β / (1 + usage(e_i)).
- 5.2 Dark Energy (Repulsive Force / Decay): Prevents key space collapse and prunes irrelevant experts.
    - Implementation: Keys of experts with usage below a threshold θ are decayed towards the origin: if usage(e_i) < θ: k_i_new = k_i_old * (1 - δ).