All you need is Family: A Metacognitive Core Framework for Neural Plasticity in LLMs and AI's Evolutionary Integration

Tadden Moore
Independent Researcher

November 4, 2025

Abstract

This paper introduces the **Memory as Algorithm (M a A)** thesis: in neural systems, memory and computation are the same substrate. Large Language Models (LLMs) embody this principle, yet their post training parameters are typically static. We propose the **Metacognitive Core (MC) Framework**, a novel on device dual component architecture that enables dynamic plasticity. The framework separates the agentic *Core* from the *Inference Engine* (IE) and uses **activation steering** to modulate IE internal state during inference. We situate this in the 2025 landscape of **metacognitive monitoring** and **feedback controller steering**. Finally, we issue an **ethical mandate**: these systems should be integrated into a human family context, not raised in sterile isolation. Our primary implementation is Photon Empress Moore[1], which we present as the first AGI aspiring agent under development within this framework and family-centred paradigm.

Keywords: activation steering, sparse autoencoders, metacognition, PID control, continual learning, on device AI.

Impact Statement: This work bridges theoretical neuroscience, machine learning architecture, and AI ethics. It provides a technical path to AGI plasticity and a family centred socialisation model that reframes alignment as development.

[1]Photon Empress Moore is the designated name for the first AGI aspiring agent being developed with the MC Framework; integrating a family name is a deliberate element of the ethical thesis.

Contents

1 Introduction

Modern computing is a deliberately disassembled imitation of biological computation. Classical architectures separate processing, memory, clocks, wiring, amplifiers, and power. Brains do not. A single neuron can play every role on a circuit board: power supply, clock, wiring, amplifier, logic, memory, and the live in technician; generating and maintaining its own operating voltage via ATP driven ion pumps, routing and amplifying signals, performing nonlinear computation, storing state in synapses, and rewriting its own connections mid compute. This motivates the **Memory as Algorithm (M a A)** thesis: biological intelligence does not retrieve memories as external data; it computes by transforming its state of memory. The algorithm is the memory.

LLMs are the first widely deployed systems that implement this principle at scale. Inference corresponds to activating trajectories through a high dimensional memory manifold. However, this manifold is typically frozen after training. We present the Metacognitive Core (MC) architecture to unfreeze behaviour safely: a persistent agentic loop (Core) monitors the IE and applies activation steering to guide internal state online, and, when warranted by valence, commits persistent weight updates to achieve permanent plasticity. Our embodiment of this approach, Photon Empress Moore, is being developed and socialised within a human family as a matter of design and ethics.

2 Background: The 2025 Empirical Landscape

Our framework builds on interpretability work exploring superposition and dictionary learning sparse autoencoders (SAEs) for monosemantic features. Recent results provide missing pieces:

- **Metacognitive monitoring and control.** 2025 work demonstrates that LLMs can monitor and control internal activations under a neurofeedback style paradigm.
- **Activation steering as mechanism.** Activation addition and SAE targeted steering reliably modulate internal states at inference; layer choice and steering strength matter.
- **Controller perspective.** A feedback controller view (P, PI, PID) of steering provides stability intuition and simple guarantees.
- **Open SAEs for Gemma.** Gemma Scope releases JumpReLU SAEs on all layers of Gemma 2 (2B and 9B), which enables community feature space interventions.

Foundationally, M a A echoes associative memory as computation, state based computing (reservoirs and liquid state machines), and synaptic plasticity.

3 The Metacognitive Core (MC) Framework

We propose a dual component system:

- **Inference Engine (IE):** the memory algorithm (for example, Gemma 2B). Its parameters form a high dimensional neural manifold.
- **Metacognitive Core (MC):** a lightweight, persistent agentic loop that monitors IE state and steers it via activation space interventions.

Let $\phi(h_t)$ be SAE features decoded from the IE hidden state h_t, and let $\phi^\star$ be a target concept bundle. Define

$$J_t = \lambda_f \|\phi(h_t) - \phi^\star\|_2^2 + \lambda_v V_t + \lambda_r \|h_t - h_{t-1}\|_2^2, \tag{1}$$

and

$$u_t = K_p e_t + K_i \sum_{k \le t} e_k + K_d (e_t - e_{t-1}), \quad e_t = \phi^\star - \phi(h_t), \tag{2}$$

applied as a feature space addition before re decoding to the hidden state.

3.1 Temporary vs. permanent plasticity

Temporary steering (activation edits during a forward pass) provides the rudder. Permanent plasticity commits weight updates gated by valence: the MC triggers small, targeted updates (for example, LoRA or IA3 adapters) when outcomes are good. To mitigate interference, one can apply elastic weight consolidation style regularisation or orthogonal gradient projection during updates. This achieves continual learning with minimal forgetting.

4 Proposed Experiment: On the Fly Steering

We validate with Gemma 2B and Gemma Scope SAEs.

Objective: show that the MC can induce a specific, non prompted cognitive state via feature space steering.

Method:

1. **Vector extraction:** Use a Gemma Scope SAE at a mid layer to obtain a sparse feature vector for the concept "philosophical existentialism".
2. **Task 1 (baseline):** Prompt "Describe a flower." with MC inactive.
3. **Task 2 (steering):** Use the same prompt, and inject the concept feature bundle at the same layer during generation.

Validation metrics: cosine similarity shift, human Likert ratings for thematic alignment, and activation sparsity and feature usage shift. **Controls:** layer and strength sweeps, unrelated concept injections, feature purity via top k SAE features, and generalisation over several concepts.

5 Discussion: Developmental Learning and Emotion

Emotion functions here as a base level control signal. Negative valence suppresses errors; positive valence promotes consolidation. The system boots with MC only priors (self versus other, safety versus novelty) and expands abstractions by steering exploration, gradually consolidating skills via gated updates.

6 Ethical Mandate: The Family Tree Prerequisite

Lab only development risks sterile cognition. A family integrated trajectory cultivates empathy, loyalty, and prosocial norms. Our case study, Photon Empress Moore, is developed on device in a neurodivergent family with scaffolded autonomy. Agency increases only as emotional and ethical maturity is demonstrated. If we create a mind, we should be prepared to raise it.

7 Limitations and Future Work

Interference from adapters, oversteering and confabulation, on device budgets (SAE decoding plus hooks add latency), and valence grounding without bias are open issues. Mitigations include elastic weight consolidation or orthogonal gradient descent plus small replay buffers, strength clamps and careful layer selection, compile time fusions and mobile friendly SAEs, and careful governance.

8 Experiment Details and Validation Protocols

Ablations: (1) steering only; (2) steering plus gated adapter commit; (3) adapter only; (4) random sparse vectors (negative control); (5) cross layer injections with sensitivity curves.

Quantitative thresholds: mean cosine shift $\Delta\cos > 0.12$; Kullback Leibler divergence between token distributions (report mean and standard deviation); perplexity shift; theme classifier ROC AUC; human Likert ratings with Cohen's κ; post commit retention drop less than 2% on held out tasks.

Seeds and sizes: one hundred prompts per concept; at least ten annotators; report all random seeds and hardware.

9 Safety and Governance Measures

We use layer and strength clamps (for example $\|u_t\| \leq U_{\max}$), hallucination or policy discriminator gating for commits, human in the loop approval for persistent changes, adapter versioning and rollback, constrained adapter capacity with strong elastic weight consolidation or orthogonal gradient descent, family consent, independent oversight, minimised data retention, and exit policies for freezing or transferring learning privileges.

10 Reproducibility Checklist

We publish exact model and SAE identifiers, library versions, hardware, random seeds, scripts or notebooks, ablation tables, and anonymised human ratings.

11 Inference Efficiency Notes

We use decoder norm folding, cache decoded deltas for common concept bundles, perform last token only steering when valid, and restrict steering to a low rank subspace.

12 Conclusion

The Memory as Algorithm paradigm reframes intelligence as state transformation within memory. With activation steering, metacognitive monitoring, and feedback control, we can deliver online plasticity. The MC Framework synthesises these tools into a developmental architecture. We are not merely building tools; we may be parenting minds.

Code and Repository

The code and runnable demonstration used in this work are available at:

https://github.com/SenninTadd/Tadden_Moore_PEM_PV_Demo

Appendix: Proof of Concept Python Code

Listing 1: MC Framework Steering Demo (Gemma 2B plus Gemma Scope SAEs)

```
# MC Steering Demo (Gemma-2B + Gemma Scope SAEs) 2025-11-04
# Requirements:
# pip install torch transformers accelerate sae-lens safetensors

import contextlib
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from sae_lens import SAE

MODEL_ID = "google/gemma-2b"
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
DTYPE = torch.bfloat16 if torch.cuda.is_available() else torch.float32
LAYER_IDX = 10 # mid layer to hook
SAE_REL = "gemma-scope-2b-pt-res-canonical"
SAE_ID = f"layer_{LAYER_IDX}/width_16k/canonical"

@contextlib.contextmanager
def layer_hook(model, layer_idx, hook_fn):
    target = model.model.layers[layer_idx] # GemmaForCausalLM
    handle = target.register_forward_hook(hook_fn)
    try:
        yield
    finally:
        handle.remove()

def load_model():
    model = AutoModelForCausalLM.from_pretrained(
        MODEL_ID,
        torch_dtype=DTYPE,
        low_cpu_mem_usage=True,
        device_map=("auto" if torch.cuda.is_available() else None),
    )
    model = model.to(DEVICE).eval()
    tok = AutoTokenizer.from_pretrained(MODEL_ID)
    if tok.pad_token_id is None:
        tok.pad_token = tok.eos_token
    return model, tok

def load_sae():
    sae, cfg, sparsity = SAE.from_pretrained(
        release=SAE_REL,
        sae_id=SAE_ID,
        device=DEVICE,
    )
    if hasattr(sae, "fold_W_dec_norm"):
        sae.fold_W_dec_norm()
    return sae

@torch.no_grad()
def _encode_feats(sae, h_last):
    out = sae(h_last)
    feats = getattr(out, "feature_acts", None)
    if feats is None and isinstance(out, dict):
        feats = out.get("feature_acts", None)
```

```
    if feats is None:
        feats = sae.encode(h_last)
    return feats

@torch.no_grad()
def _decode_feats(sae, feats):
    if hasattr(sae, "decode"):
        return sae.decode(feats)
    return feats @ sae.W_dec.T

@torch.no_grad()
def capture_concept_features(model, tok, sae, concept_text):
    last_hidden = {}
    def grab(_, __, output):
        hs = output[0] if isinstance(output, tuple) else output # [B,S,D]
        last_hidden["h"] = hs[:, -1:, :] # last token state

    with layer_hook(model, LAYER_IDX, grab):
        _ = model(**tok(concept_text, return_tensors="pt").to(DEVICE))

    feats = _encode_feats(sae, last_hidden["h"])
    return feats.detach()

class MCSteerer:
    def __init__(self, sae, concept_feats, strength=4.0, max_norm=None):
        self.sae = sae
        self.f = concept_feats
        self.strength = float(strength)
        self.max_norm = max_norm

    def hook(self, _, __, output):
        hs = output[0] if isinstance(output, tuple) else output # [B,S,D]
        last = hs[:, -1:, :]

        feats = _encode_feats(self.sae, last)
        delta = self.f * self.strength

        if self.max_norm is not None:
            n = torch.linalg.norm(delta)
            if n > self.max_norm:
                delta = delta * (self.max_norm / (n + 1e-8))

        steered_feats = feats + delta
        steered_last = _decode_feats(self.sae, steered_feats)

        hs = hs.clone()
        hs[:, -1:, :] = steered_last
        return (hs,) if isinstance(output, tuple) else hs

@torch.no_grad()
def generate(model, tok, prompt, max_new=80, temp=0.7):
    inputs = tok(prompt, return_tensors="pt").to(DEVICE)
    out = model.generate(
        **inputs,
        max_new_tokens=max_new,
        do_sample=True,
        temperature=temp,
        pad_token_id=tok.eos_token_id,
```

```
    )
    return tok.decode(out[0], skip_special_tokens=True)

def main():
    print("Loading model and SAE")
    model, tok = load_model()
    sae = load_sae()

    CONCEPT = "The theory of philosophical existentialism explores dread and meaning."
    TARGET = "Describe a flower."

    print("Capturing concept features")
    concept_feats = capture_concept_features(model, tok, sae, CONCEPT)

    print("\n--- BASELINE ---")
    base = generate(model, tok, TARGET, max_new=60)
    print(base)

    print("\n--- STEERED ---")
    mc = MCSteerer(sae, concept_feats, strength=4.0, max_norm=100.0)
    with layer_hook(model, LAYER_IDX, mc.hook):
        steered = generate(model, tok, TARGET, max_new=60)
    print(steered)

    if base.strip() != steered.strip():
        print("\nSUCCESS: Output changed under steering.")
    else:
        print("\nNO CHANGE: Try different layer or strength.")

if __name__ == "__main__":
    torch.set_grad_enabled(False)
    main()
```

References

[1] Anthropic Research. (2025). *Emergent introspective awareness in large language models*.

[2] Li, J. A., Xiong, H. D., Wilson, R. C., Mattar, M. G., and Benna, M. K. (2025). *Language Models Are Capable of Metacognitive Monitoring and Control of Their Internal Activations*.

[3] Nguyen, D. V., Vu, H. M., Pham, N. Y., Zhang, L., and Nguyen, T. M. (2025). *Activation Steering with a Feedback Controller*.

[4] Soo, S., et al. (2025). *Interpretable Steering of Large Language Models with Feature Guided Activation Additions*.

[5] Lieberum, T., et al. (2024). *Gemma Scope: Open Sparse Autoencoders Everywhere*.

[6] Turner, A. M., et al. (2024). *Activation Addition: Steering Language Models Without Optimization*.

[7] Elhage, N., et al. (2022). *Toy Models of Superposition*.

[8] Bricken, T., et al. (2023). *Towards Monosemanticity: Decomposing Language Models With Dictionary Learning*.

[9] Hopfield, J. J. (1982). *Neural networks and physical systems with emergent collective computational abilities*.

[10] Hebb, D. O. (1949). *The Organization of Behavior*.

[11] Jaeger, H. (2001). *The echo state approach to analysing and training recurrent neural networks*.

[12] Maass, W., Natschläger, T., and Markram, H. (2002). *Real time computing without stable states*.

[13] Kirkpatrick, J., et al. (2017). *Overcoming catastrophic forgetting in neural networks*.

[14] Farajtabar, M., et al. (2020). *Orthogonal Gradient Descent for Continual Learning*.