oczy: A Plastic World Model Agent

kinoSoft Research Lab

June 2026

Abstract

oczy is a research prototype aimed at the class of agents that learn by changing rather than retrieving. The target architecture turns experience into fast state, fast state into fast weights, fast weights into slow priors, and finally discards the raw trace after consolidation. We describe the prototype's organ design, the three engineering goals that must be met before the metabolism closes, and the experimental findings collected so far.

1. The retrieval bottleneck

Most current language-agent memory systems store experience as retrievable content: a chat log, a vector database, or a set of reasoning traces. The agent answers by recalling the right note and pasting it into a prompt. This works for lookup, but it is not learning. If the stored note is missing, the agent reverts to its base behavior. If many notes conflict, it must choose among them at generation time. And if an old fact should be eclipsed by a correction, the system still carries the original trace.

What we want instead is *changed dynamics*. A correction should alter the agent so that future behavior is different even when the original utterance is no longer in context. The memory should live in the parameters and state of the agent, not in an external cache. This is the premise of the Plastic World Model Agent.

2. Target architecture

The desired system is a continuum:

```
activation / context

fast state

fast weights

neural memory

slow weights / priors
```

not a pipeline from chat log through a vector store back to a prompt. Formally, let:

θ slow weights / long-term priors,
ϕ_t fast weights / session adaptation,
m_t neural memory state,
z_t identity/task latent state,
x_t current input,
c_t correction or outcome signal.

The forward pass, error, and updates are then:

$$
\begin{aligned}
h_t &= \mathrm{Core}(x_t, \theta, \phi_t, m_t, z_t), \\
\hat{y}_t &= \mathrm{Decode}(h_t), \\
\mathrm{error}_t &= \mathrm{PredictionError}(\hat{y}_t, c_t), \\
\phi_{t+1} &= \mathrm{FastUpdate}(\phi_t, \mathrm{error}_t), \\
m_{t+1} &= \mathrm{MemoryUpdate}(m_t, h_t, \mathrm{error}_t), \\
z_{t+1} &= \mathrm{IdentityUpdate}(z_t, \mathrm{error}_t), \\
\theta &\leftarrow \mathrm{SlowConsolidation}(\theta, \mathrm{replay}(m), \mathrm{constraints}).
\end{aligned}
$$

The important property is that the raw trace is used once, then compressed, replayed, and forgotten. What remains is a changed model.

3. The tower: six organs

The current prototype decomposes the agent into six organs, each responsible for one gate in the metabolism. Table 1 lists their roles, current inputs, and target inputs.

Organ	Floor	Today consumes	Should consume
Plastic Cortex	Belfry	Random projection; `raw_hidden` steering	SVD-aligned hidden steering
Neural Hippocampus	Scriptorium	Episode dicts, hash embeddings	Cortex hidden tensors
World-Model Critic	Observatory	Hand-built string features	`warm_cold_drift` from cortex
Identity Hypernetwork	Throne	`concept_scores` dict	Cortex state deltas
Skill Immune Cortex	Gatehouse	Keyword trigger strings	Anti-direction KV-poison entries
Experience Autoencoder	Cellar	Returns Δz	Δz via `cortex.train_step`

At present the LM still sits on the answer path, while the cortex writes label strings. The intended inversion places the cortex at the live center and reduces the LM to a perception/articulation organ on the periphery.

4. Implementation goals

Table 2 lists the three engineering goals that reverse the current inversion. Goal 1 is the most critical: without a steering surface into the LM, no metabolism result is observable in generated text. Goal 2 can start in parallel once Goal 1's binding surface is stable. Goal 3 closes the loop through the remaining organs.

Goal	Description	Status
1	LM-side steering binding via a reserved KV slot or proxy	Proxy
2	Hidden-state extraction at layer L for real cortex input	Under investigation
3	Upgrade five metabolism organs to consume tensor inputs	Pending

Goal 1: steering binding

The cortex emits per-layer control vectors through `emit_cvec`. In the `llama-cpp-python` binding these are applied as residual bias, which shifts *posture* such as domain or framing but not an arbitrary individual token. A reserved KV slot would let the cortex inject factual content at a fixed sequence position.

Direct KV-slot writing is not yet exposed by the binding. The practical proxy is a fixed text prefix prepended to every generation prompt. A probe on 2026-06-25 showed that this proxy achieves exact-token uptake ("vertical" appeared) where the residual control vector only produced a domain shift. The current driver therefore exposes two complementary surfaces:

- residual control vectors for posture and framing, and
- reserved positions for exact fact recall.

Long term, the reserved position should be implemented as a prefilled KV cache chunk once the binding supports it, avoiding the cost of re-encoding text tokens on every call.

Goal 2: hidden-state extraction

The cortex's warm path requires the LM residual at a chosen layer L, not the embedding-layer output and not a synthetic placeholder. The candidate paths under investigation are `Llama._internal_state`, a twin eval with an activation callback, and, as a fallback, a fork of the binding with a hookable eval path. Done when `driver.peek_layer(prompt, layer_idx)` returns a non-trivial d_{emb} array and training on real hiddens produces visibly different `warm_state` trajectories.

Goal 3: organ upgrades

The five metabolism organs still consume strings and parsed episode dictionaries. The cortex cannot metabolize in isolation; the fast state must flow through the hippocampus, critic, identity, immune, and autoencoder. Done when a correction through `CortexAgent` produces visible `cold_state` drift after `consolidate()`, when repeated corrections compound rather than overwrite, and when Stage 2 of the organism curriculum (scope control) becomes tractable.

5. Experimental findings

5.1 Raw-hidden steering collapses the binary cliff

On 2026-06-24 we implemented `steering_mode="raw_hidden"`. Instead of projecting the last-correction hidden through a random `proj_c`, it broadcasts the L2-normalised raw hidden across layers scaled by the warm amplitude. On `LFM2.5-1.2B-Instruct-Q4_K_M`, `articulate_scale=0.001` yields clean on-manifold leaning toward the corrected sense; below 0.0005 the signal vanishes and at 0.005 it breaks into token-repetition. The previous random-projection behaviour—no effect below scale 30, then collapse—has been replaced by a usable linear regime.

5.2 Prefix steering achieves exact-token uptake

On 2026-06-25 a fixed prefix such as "In this codebase, profile means business vertical." was prepended before generation. In that condition the token "vertical" appeared reliably, whereas the residual control vector only shifted the answer toward the commercial domain. This confirms that a reserved-position signal is the correct surface for fact recall, even if its first implementation is text rather than a KV write.

5.3 Architecture review recovers the upper bound

A 2026-06-21 review found that `NeuralHippocampus` scoring 1.000 above the Oracle at 0.844 was a measurement artifact. The hippocampus score measured internal bookkeeping rather than behavioural uptake, and the Oracle score suffered from a metric-direction bug. After fixing these, `NeuralHippocampus` still ranked first on internal mechanics, the Oracle ranked third as a proper upper bound, and the baseline ordering was restored.

The same pass also exposed a schema drift between `correction` and `corrected_answer` that was silently killing replay in the ranker.

5.4 Organism curriculum

The six-stage curriculum probes grounding, transfer, scope control, dialog, consolidation, and cross-domain behaviour. Raw runs absorb Stage 0, Stage 1, and Stage 4 cleanly; Stage 2 fails because the toy `PlasticCortex` backend cannot hold two senses per token, and Stage 3 and Stage 5 remain partial. LM-mediated runs parse roughly 42–67% of lessons depending on curriculum wording but still reach 92% absorption through the raw fallback path.

6. Evaluation roadmap

The right benchmark is not a chat corpus; it is a correction-to-competence benchmark. Each task should require an initial plausible mistake, a user correction, a corrected answer, and later probes for transfer, scope control, and forgetting. Useful metrics include:

- correction uptake latency (turns until behavior changes),
- one-shot transfer to related cases,
- scope control (does the correction stay in the right domain?),
- forgetting of unrelated old competence,
- compression (memory growth per correction),
- consolidation (whether raw traces can be dropped without loss),
- identity drift under many corrections.

The single summary metric we care about is

$$\frac{\text{behavior delta}}{\text{bytes of persistent memory}},$$

which measures how much learned behavior is gained per unit of stored state.

7. Conclusion

oczy is not an LLM with an external memory layer. It is an attempt to build an agent whose memory is changed weights and state. The climb is currently at the binding layer: once the cortex can write into the LM and read real activations back, the organ upgrades become observable experiments. Until that loop closes, the architecture remains a scaffold with an answer path that still runs through the LM rather than through the cortex.

Canonical source. The cortex contract tests live at `plastic-cortex/src/plastic_cortex/kv_cortex.py` (9/9 passing as of 2026-06-23). This PDF is a snapshot; the latest plans, logs, and code are in the `oczy` repository.