OpenClawBrain v12.2.6+: Shadow Routing with QTsim Confidence Mixing and Unified Policy Learning

Jonathan Gu

March 2, 2026

Version: v12.2.6+

1 Abstract

OpenClawBrain is a retrieval-routing system for long-lived agents built around a strict hot/cold path split. The hot path is local and bounded for latency-critical routing; the cold path runs asynchronous learning and structure updates. We define a *turn* as a single LLM API call. End-to-end cost is $C = \sum_{i=1}^{T} \text{tokens}_i \cdot \text{price}_i$, and the system targets lower T (fewer calls per user-visible request) and lower per-call token counts on the hot path. The current system uses two route signals: a structural prior (graph_prior) and a query-time compatibility signal (QTsim). A confidence mixer combines those signals using uncertainty features, including entropy and margin, to decide how much weight each source gets per decision. Learning is performed through teacher distillation plus policy-gradient updates under a reward-source hierarchy. Human corrections have highest authority, then self-learning outcomes, then harvested signals, then high-volume teacher labels. This release also starts RL-native maintenance of graph structure: Phase 2a is shipped and follow-on phases cover connect/split/merge actions under policy control.

2 1. System Overview

OpenClawBrain is designed around a strict split:

- **Hot path:** local embedding retrieval and bounded traversal only.
- **Cold path:** asynchronous learning and structure updates.

We define a *turn* as a single LLM API call. Total cost can be written as

$$C = \sum_{i=1}^{T} \text{tokens}_i \cdot \text{price}_i$$

and OpenClawBrain reduces T and per-call token counts by keeping routing local and context bounded on the hot path, while allowing optional teacher calls only in the cold path.

1.1 Routing Inputs

At each route decision over candidate targets $\mathcal{A}(s)$ from state s:

- $\text{graph\_prior}(a \mid s)$ captures persistent graph structure.
- $\text{QTsim}(a \mid q, s)$ captures query-conditioned fit.

1.2 Confidence Mixing

The router computes a query-local mix coefficient $\lambda \in [0,1]$ from confidence features. Two required features are:

- **Entropy** of QTsim over candidates (high entropy means low confidence).
- **Margin** between top-1 and top-2 QTsim scores (low margin means ambiguity).

A simple form is:

$$\lambda = \sigma\big(\alpha_0 + \alpha_1 \,\mathrm{margin}_{\mathrm{QTsim}} - \alpha_2 \, H(\mathrm{QTsim})\big)$$
$$\pi(a \mid q, s) = \lambda \,\mathrm{QTsim}(a \mid q, s) + (1 - \lambda)\,\mathrm{graph\_prior}(a \mid s)$$

where σ is a logistic gate and H is entropy. When QTsim is sharp and separated, λ increases. When QTsim is uncertain, the policy falls back toward graph prior.

3 2. Learning Loop: Distillation + Policy Gradient

The learning loop combines distillation-style supervision with policy-gradient refinement.

2.1 Teacher Distillation

Offline replay samples route decisions and asks a teacher policy for preferred actions (or distributions). Distillation improves route quality quickly and provides dense supervision for states with sparse human labels.

2.2 Policy-Gradient Update

Given route trajectory $\tau = (s_0, a_0, \ldots, s_T, a_T)$ and scalar return $R(\tau)$:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} (R(\tau) - b_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

where b_t is a baseline. Distillation targets and reward-weighted gradients are both applied; the exact blend is implementation-defined per run mode.

2.3 Reward-Source Hierarchy

The system resolves conflicting supervision by ordered authority:

1. Human correction/approval (highest authority)
2. Self-learning outcomes from deployed agents
3. Harvested weak signals
4. Teacher labels from async replay (lowest per-sample authority, highest volume)

This hierarchy constrains updates so high-authority signals can override lower-level noise.

4 3. RL-Native Maintain (Structure Learning)

OpenClawBrain treats structure maintenance as a control problem over graph actions.

3.1 Phase 2a (Shipped)

Phase 2a ships RL-native maintenance hooks that integrate policy outcomes with structure scoring and safe-apply gates. In practice, this means structure candidates are generated, scored with policy-derived signals, and applied only when guardrails pass.

3.2 Roadmap: connect/split/merge

Planned phases extend action space with explicit graph edits:

- **Connect:** create/reweight candidate edges when repeated successful transitions lack direct structure.
- **Split:** divide overloaded nodes when policy confidence or error patterns indicate topic collapse.
- **Merge:** combine nodes with persistent redundancy and equivalent downstream behavior.

Each action class is expected to use conservative thresholds, rollback support, and offline shadow evaluation before broad enablement.

5 4. Evaluation: Industry Baselines + Ablations

Evaluation is organized around industry-standard baselines and fully reproducible artifacts.

4.1 Baselines (Why These Comparisons)

We compare against three industry baselines plus the learned router:

- **Vector top-k** (`vector_only`): plain embedding retrieval, no traversal.
- **Top-k + reranker** (`edge_sim_legacy`): deterministic re-ranking with `route_mode=edge+sim`.
- **Pointer-chasing** (`edge_sim_legacy` with multi-hop traversal): iterative retrieval without learned routing.
- **OpenClawBrain learned** (`learned`) with ablation bounds `graph_prior_only` and `qtsim_only`.

These modes are exposed in the evaluation harness in the main `openclawbrain` repo.

4.2 Core Metrics

- **Reward / accuracy**: task success or correctness rate (dataset dependent).
- **% oracle gap closed**: fraction of the oracle gap closed over training epochs (expert-regions simulation).
- **Latency**: p50/p95 end-to-end query time from the eval harness.
- **Tool calls, tokens, cost**: measured in downstream agent runs; report as `TBD` if not available.

4.3 Figures (SVGs in `figures/eval`)

SVG sources live in `figures/eval/*.svg`; the PDF renders the PNG versions below.

4.4 Reporting Template (Do Not Invent Numbers)

All unreproduced results must be marked `TBD` and tied to an output file path.

Setting	Reward/Acc	% Oracle Gap	Latency p50/p95	Tool Calls	Tokens	Cost
Vector top-k (expert-regions sim)	0.878629 / 0.8920	TBD	TBD	TBD	TBD	TBD
Top-k + reranker	0.879762 / 0.8935	TBD	TBD	TBD	TBD	TBD
Pointer-chasing (graph prior only)	0.555147 / 0.5290	TBD	TBD	TBD	TBD	TBD
OpenClawBrain learned (mixed)	0.935126 / 0.9680	96.74%	TBD	TBD	TBD	TBD

Table 1: Expert-regions simulation baselines from reproducible runs. Each `TBD` must cite commit SHA, command, and artifact path (e.g., `/tmp/ocb_eval.json`).

4.5 10-Seed Ground-Zero Proof (March 2026)

The ground-zero harness scales evaluation to 800 queries per seed across 10 independent seeds, with relation drift injected mid-run. Table 2 shows the core comparison.

Baseline	Accuracy	Context Used	Traversal Cost	Win Rate vs `full_brain`
`full_brain`	**0.9723**	800	800	—
`vector_rag_rerank`	0.8901	4 000	6 400	1/10
`vector_rag`	0.7966	800	800	0/10
`heuristic_stateful`	0.7943	800	800	0/10

Table 2: Ground-zero 10-seed proof (`proof_10seed_20260306`). `full_brain` wins 10/10 seeds vs `vector_rag` and `heuristic_stateful`, 9/10 vs `vector_rag_rerank`. The nearest competitor uses 5× the context and 8× the traversal cost for lower accuracy.

Worked example: E048 :: E001. The ground-truth relation changes from `manages` to `depends_on` at step 22. `full_brain` tracks the change at steps 22 and 38; `heuristic_stateful` stays stale; the vector baselines stay wrong or lag. This is bounded benchmark evidence—mechanism proof, not a claim of recorded-session, shadow, or online production superiority.

4.6 Reproduction Commands (Expected Output Paths)

Commands (from `openclawbrain` repo) and expected outputs:

```
cd /path/to/openclawbrain
python examples/eval/run_eval.py \
  --state /path/to/state.json \
  --queries /path/to/queries.jsonl \
  --modes vector_only,edge_sim_legacy,graph_prior_only,qtsim_only,learned \
  --output /tmp/ocb_eval.json \
  --print-per-query

python examples/eval/simulate_expert_regions.py --output-dir /tmp/ocb_expert_regions
python examples/eval/simulate_two_cluster_routing.py --output-dir /tmp/ocb_two_cluster
```

```
python3 /path/to/openclawbrain-site/scripts/plot_eval.py \
  --inputs /tmp/ocb_expert_regions /tmp/ocb_two_cluster \
  --out figures/eval
```

Expected outputs include `/tmp/ocb_eval.json`, `/tmp/ocb_expert_regions/simulation_curve.csv`, and regenerated SVGs in `figures/eval/`.

6 5. Operational Notes

OpenClawBrain now supports local embeddings out-of-the-box for operator-first deployment. This enables deterministic bootstrap and private/offline-friendly operation while keeping optional teacher workflows in asynchronous jobs.

7 6. Conclusion

The v12.2.6+ system is defined by four commitments: local hot-path retrieval, QTsim+graph-prior confidence mixing, distillation plus policy-gradient learning, and RL-native structure maintenance. Performance claims should be published only from reproducible evaluation runs with explicit provenance.

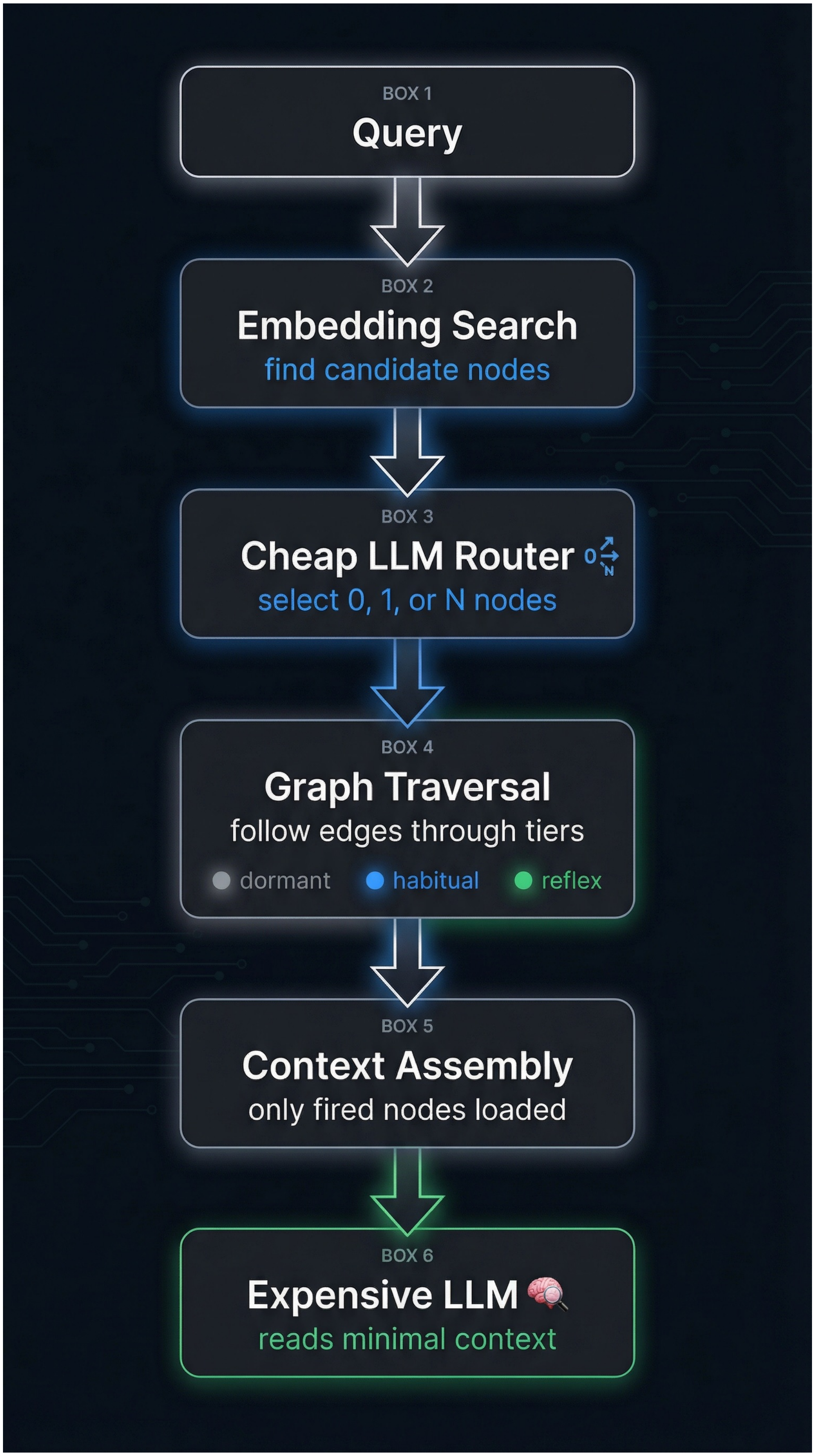

Figure 1: System diagram: latency-critical hot path stays local and bounded, while shadow replay and policy updates run asynchronously in the cold path.

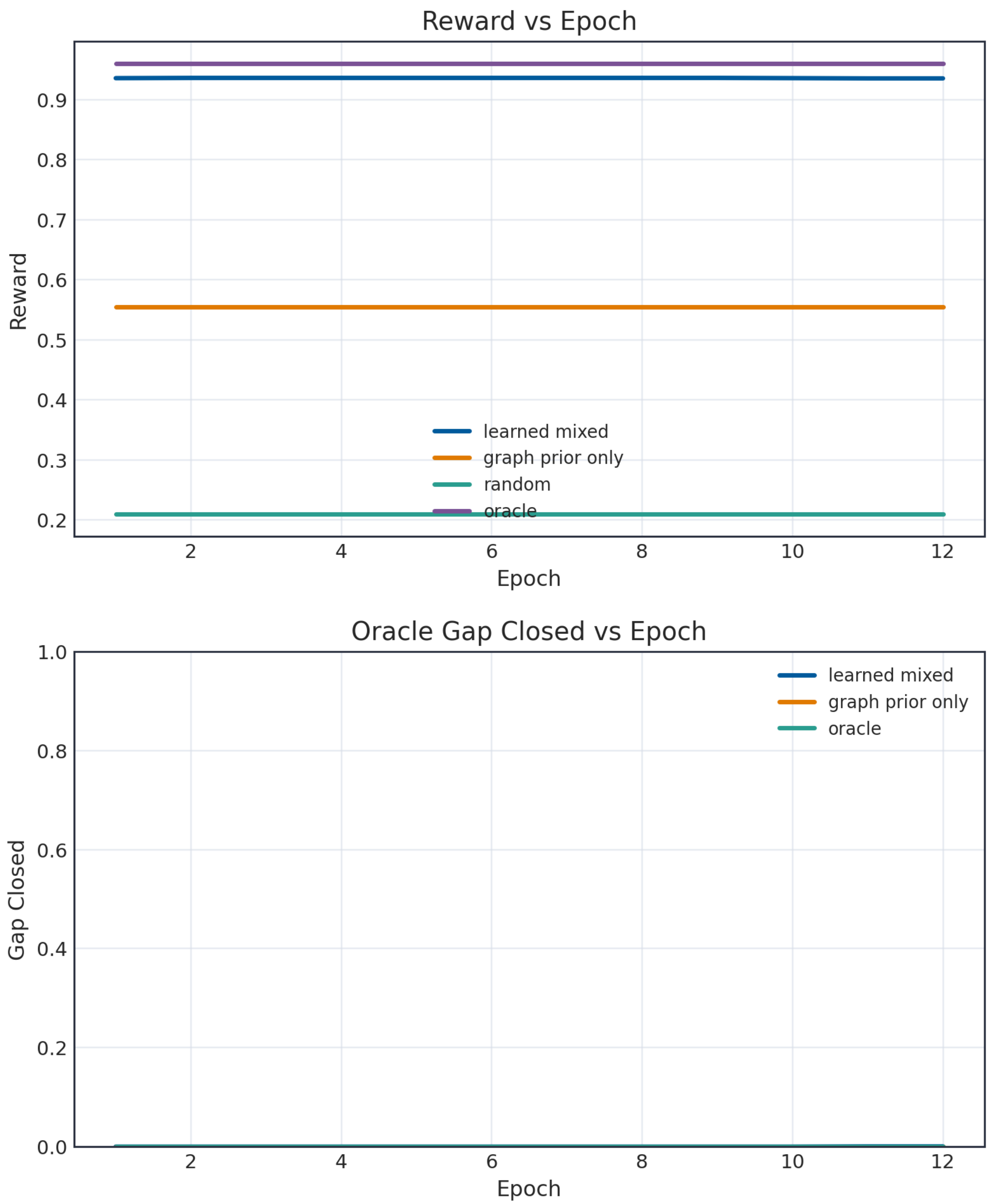

Figure 2: Expert-regions simulation learning behavior: reward trajectory and fraction of oracle gap closed across training epochs.

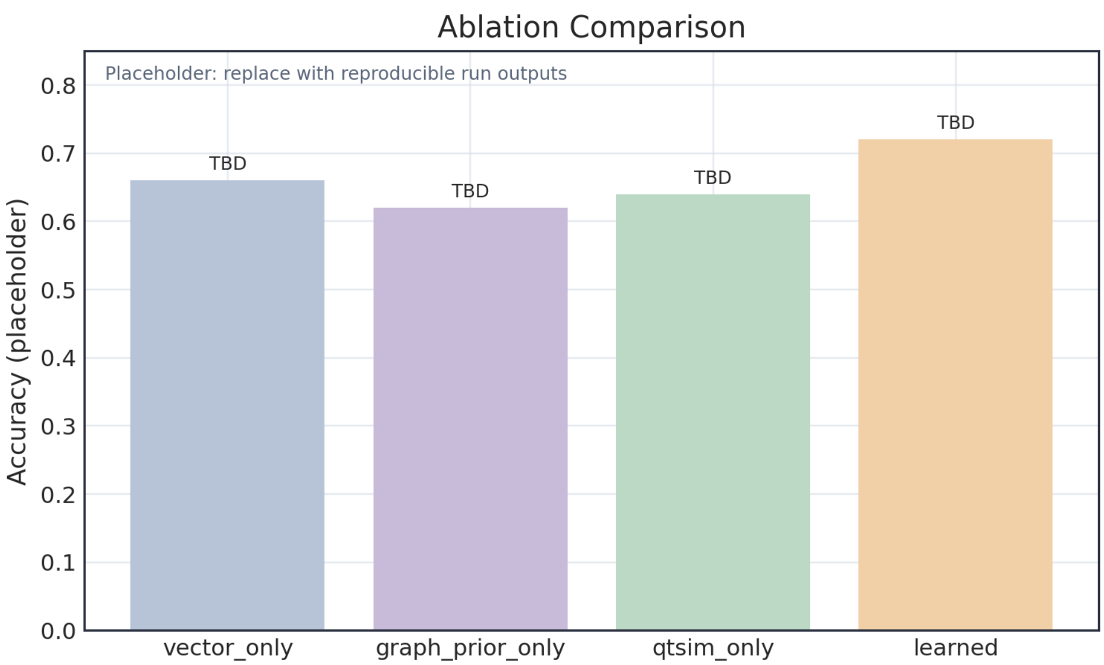

Figure 3: Ablation comparison across `vector_only`, `graph_prior_only`, `qtsim_only`, and learned routing.