Stop Picking Routes by Vibes

Fractal Cell Energy: A Biological Architecture for Route Selection over an Agent's Tool Library

Lewis Tham
Unbrowse AI
lewis@unbrowse.ai

Cayden Chik
Unbrowse AI
cayden@unbrowse.ai

June 2026

Abstract

An autonomous agent accumulates a library of callable routes — learned first-party API endpoints, each answering some class of intent. The runtime question is selection: given a user intent, which route should fire? We answer it the way biology already did, which is mildly embarrassing for the field. Each route is a *cell* that carries a scalar *energy* — the negative coherence between the intent and what the route is for — and trains that energy every time it fires: a reward-weighted running-mean prototype ($O(1)$ per firing, West 1979) that folds toward the context on a good outcome and away on a bad one. Selection is then not a decision but a measurement — rank the library by ascending energy $E(\text{intent}, \text{route})$, fire the lowest. The operation is scale-free: it composes unchanged from the single neuron up through the atom→cell→tissue→organ fractal, so select(select) = select all the way down — which is either a deep structural fact or the only trick we know, and the experiments do not require us to decide. A content-addressed cache makes a repeated selection a lookup, on the principle that a cell which already fired should not have to think again. We report only reproduced results. On discrete-structure tasks the method wins: route ranking reaches **R@1** = 0.0488, **4.6**× a keyword baseline; access-pattern type classification reaches **0.71–0.81 accuracy**, **≈2.1**× baseline across two seeds; a closed-loop learned ranker reaches a cold-cell **AUC of 0.750** against 0.5 chance; the runtime-embedded head scores **0.83 warm / 0.64 cold**; and a content-addressed cache yields a **92× mean speedup** on a reuse micro-benchmark. We are equally explicit about where the method gives nothing, because a ranker that wins everywhere is usually a ranker that has not been measured carefully: energy-reranking a free-form language model's own generations is indistinguishable from random (lift +0.000), and on a public 101-route retrieval set a plain keyword baseline beats the energy ranker. The honest boundary is the contribution: a learned energy plus a content-addressed cache wins on *discrete structure* — route retrieval, type classification, cache reuse — and contributes nothing to free-form prose generation.

1 Introduction

Once an agent learns callable routes from real usage, it holds a growing tool library: each route is a typed, replayable first-party endpoint that answers some class of intent. The hard runtime problem is no longer discovery but *selection* — given an incoming intent and dozens or hundreds of candidate routes, pick the one that actually answers it. A naive keyword or embedding match treats the library as a flat lexical index and ignores the structured signal in how a route was used and what it returned.

We frame selection as energy-based learning [1]. A learned energy head maps an *(intent, route)* pair to a scalar *energy* — low energy means the route is compatible with the intent, high energy

means it is not. The selector embeds the intent, scores every candidate route, ranks the library by ascending energy, and fires the lowest-energy route. The head is trained discriminatively from observed *(intent, route, outcome)* triples: routes that succeeded for an intent are pushed to low energy, routes that did not are pushed up. The trained head is small enough to embed directly in the runtime, so ranking is an inference pass, and a content-addressed cache stores each result under the hash of its content so a repeated selection resolves without recomputing.

We make one claim and bound it carefully. The method is a *discrete-structure ranker*: it wins where candidates carry separable structure — distinct routes, distinct access-pattern types — and it wins by a wide margin there. It is not a generative model and adds no signal to free-form text. This paper reports both sides as reproduced measurements, and the negative side is stated as plainly as the positive. Our contributions:

1. An energy-based selector over an agent's route/tool library, trained from observed outcome triples and embedded into the runtime for inference-time ranking.
2. A reproduced evaluation on discrete-structure tasks — route ranking, access-pattern type classification, a closed-loop learned ranker, a runtime-embedded head, and content-addressed cache reuse — each with a runnable gate.
3. An explicit negative-results section establishing the method's boundary: it does nothing for free-form generation, and a keyword baseline can beat it on a public retrieval set. The boundary is the result.

2 Related work

2.1 Energy-based and joint-embedding models

Energy-based models score the compatibility of an input pair with a single scalar and select by minimising that scalar, rather than normalising a probability over a large output space [1]. Joint-embedding predictive architectures extend the idea to representation learning: predict the latent of a target from the latent of a context and penalise the gap in embedding space [2]. We borrow the scoring-and-ranking shape — score each candidate, rank by energy, pick the lowest — and apply it to a concrete, finite selection problem (which route fires) rather than to representation pre-training.

2.2 Retrieval and learned ranking

Dense retrieval embeds queries and documents into a shared space and ranks by similarity [3]; learning-to-rank optimises an ordering objective over candidates [4]. Our setting differs in the training signal: candidates are *routes*, and the supervision is the observed *outcome* of firing a route for an intent, not a relevance label. The energy head is a discriminative ranker over that outcome signal.

2.3 Tool selection for agents

Tool-use work teaches a model *when* to call a tool [5] and to emit correct calls across many declared APIs [6]. These rank tools inside the language model's own forward pass. We separate selection from generation: a dedicated energy head ranks the route library, and the language model is left to do what it is good at. The negative-results section below is precisely the boundary between these two responsibilities.

3 Method

3.1 The selection problem

Let an intent q arrive against a library $\mathcal{R} = \{r_1, \ldots, r_n\}$ of candidate routes. A learned energy head E_θ scores each pair:

$$s_i \;=\; E_\theta(q, r_i), \tag{1}$$

and the selector returns the lowest-energy route,

$$r^\star \;=\; \arg\min_{r_i \in \mathcal{R}} E_\theta(q, r_i), \tag{2}$$

or, for a shortlist, the library sorted by ascending s_i. Low energy encodes "this route answers this intent."

3.2 Training the energy head

The head is trained from observed *(intent, route, outcome)* triples logged at runtime. For an intent, the route that successfully answered it is the positive; a sampled set of other library routes are negatives. The objective pushes the positive's energy below the negatives' by a margin, so that Eq. (2) selects the route that historically worked. Because the supervision is the real outcome of firing a route, the head improves as the agent accumulates usage — the ranker is closed-loop.

3.3 Embedding the head in the runtime

The trained head is small and is loaded directly into the live selection path, so ranking the library is one inference pass per resolution. A content-addressed cache sits in front of execution: each intermediate result is stored under the hash of its content, so a second identical selection resolves to a cached lookup instead of re-scoring and re-firing. Selection cost is therefore paid once per distinct intent and amortised over every repeat.

3.4 The self-similar shape

The same score-rank-select shape recurs at more than one scale. Selecting *which* route fires is one energy ranking; classifying a route's *access-pattern type* (e.g. anchor, server-rendered, structured-query) is the same energy ranking over a small label set. A single learned-energy mechanism serves both, which is why both appear in the evaluation.

4 Evaluation

We report only results backed by a runnable witness that re-runs green. Each row of Table 1 names its gate; the appendix lists the reproduce commands. Both the energy path and each baseline are scored on the same held-out data.

4.1 Discrete-structure results (wins)

Route ranking. Ranking the true route against 99 distractors for a real intent, the energy head reaches **R@1** $= 0.0488$ **versus** 0.0106 **for a keyword baseline — a 4.6× improvement**. The absolute number is small because the task is severe (1-in-100 with real, near-duplicate distractors); the relevant quantity is the multiple over the lexical baseline, and the learned energy more than quadruples it.

Task	Metric	Energy	Baseline
Route ranking	R@1 (1 true / 99 distractors)	**0.0488**	0.0106
Access-pattern type	accuracy (2 seeds)	**0.71–0.81**	0.32–0.39
Closed-loop ranker	cold-cell AUC	**0.750**	0.500
Runtime-embedded head	warm / cold score	**0.83 / 0.64**	—
Content-addressed cache	reuse speedup	**92**×	1×

Table 1: Reproduced discrete-structure results. Each row re-runs green under its gate (Appendix A).

Access-pattern type classification. Energy-ranking a route's access-pattern over a 7-class label set reaches **accuracy** 0.71–0.81 **across two seeds,** ≈**2.1**× **a** 0.32–0.39 **baseline**. The structure here is discrete and separable, which is exactly the regime where the energy ranker is strong.

Closed-loop learned ranker. The closed-loop variant — where the head trains on accumulated outcomes and then ranks unseen cells — reaches a **cold-cell AUC of** 0.750 **against** 0.5 **chance**, confirming that a learned ranker ships and generalises to cold cells, not just memorises warm ones.

Runtime-embedded head. The head embedded into the live runtime scores 0.83 **warm and** 0.64 **cold**, confirming that the trained selector loads and ranks inside the serving path, not only in an offline harness.

Content-addressed cache reuse. The content-addressed cache yields a **92**× **mean speedup** on a reuse micro-benchmark that isolates a warm cached selection against a cold recomputation. This is a mechanism demonstration: it shows the categorical warm-versus-cold gap, not an end-to-end task number.

4.2 Honest negatives — where the method gives nothing

The boundary of the method is as important as its wins, and we report it as measured failure, not omission. Table 2 lists the negatives.

Task	Result	Reading
Rerank a language model's generations	lift +0.000	= random
Public 101-route retrieval set	keyword wins	baseline beats energy

Table 2: Reproduced negatives. The method adds no signal to free-form generation, and is not universally better than a lexical baseline on retrieval.

Free-form generation: nothing. Using the energy head to rerank a free-form language model's own candidate generations produces a **lift of** +0.000 — **indistinguishable from random selection**. The energy carries no signal about free-form answer quality. This is the sharp edge of the method: it ranks *discrete candidates with separable structure*, and a model's prose continuations are neither.

Public retrieval: a keyword baseline can win. On a public 101-route retrieval set, a **plain keyword baseline beats the energy ranker**. We do not hide this behind the favourable route-ranking number above. The energy ranker's advantage is not universal; on some retrieval distributions a lexical match is simply better, and the method should not be deployed as if it dominates everywhere.

The boundary, stated plainly. Taken together, the wins and the negatives draw one line: *a learned energy plus a content-addressed cache wins on discrete structure* — route retrieval, type classification, cache reuse — *and contributes nothing to free-form prose generation.* The selector picks which route fires; it does not, and should not, replace the language model that writes text. This boundary is the honest core of the result, and we state it rather than engineer around it.

5 Discussion

The result is deliberately narrow. Energy-based selection is the right tool when the candidate set is finite, the candidates carry separable structure, and there is an observed outcome signal to train on — which is exactly the situation of a runtime choosing among an agent's learned routes. It is the wrong tool when the "candidates" are open-ended text continuations, because the energy has no separable structure to score; there, the measured lift is zero. Reporting the zero is the point: the inverse of optimising a metric until it stops meaning anything is naming, precisely, where the metric goes flat.

Two consequences follow. First, selection and generation are best kept as separate responsibilities: a discriminative energy head for *which route*, a generative model for *what to say*. Second, the absolute route-ranking number is a floor, not a ceiling — it is a 1-in-100 task with real distractors, and the load-bearing quantity is the 4.6× multiple over the lexical baseline, which the cache then makes cheap to serve repeatedly.

6 Conclusion

We presented energy-based route ranking: a learned energy head that scores *(intent, route)* compatibility, ranks an agent's tool library by ascending energy, and fires the lowest-energy match, trained from observed outcome triples and embedded into the runtime with a content-addressed cache. On discrete-structure tasks the method wins and reproduces — route ranking at 4.6× a keyword baseline, type classification at ≈2.1×, a closed-loop ranker at AUC 0.750, a runtime head at 0.83/0.64, and a 92× cache reuse speedup. On free-form generation it wins nothing (lift +0.000), and on at least one public retrieval set a keyword baseline beats it. The contribution is the line between those two halves: the method is a discrete-structure selector, and naming where it stops is what makes the wins trustworthy.

A Reproducibility

Every reported number re-runs from a single command. The full set runs with `bash bench/energy/reproduce-all`. Individual gates:

- Route ranking (R@1, 4.6×): `bash bench/energy/route-ranking-gate.sh`.
- Access-pattern type classification (≈2.1×, 2 seeds): `bash bench/energy/intent-type-gate.sh`.

- Closed-loop learned ranker (cold-cell AUC 0.750): `bash bench/ebm-closed-loop-gate.sh`.
- Runtime-embedded head (warm 0.83 / cold 0.64): `bash bench/ebm-runtime-ship-gate.sh`.
- Content-addressed cache reuse (92×): `python3 paper/reference/bench/bench_reuse.py`.

Each gate exits 0 only when its measured number clears its threshold, so a regression fails honestly rather than printing a stale figure. The negatives are recorded in the same accumulating ledger as the wins, so the boundary is part of the reproduced record, not an aside.

References

[1] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang. *A Tutorial on Energy-Based Learning.* In *Predicting Structured Data*, MIT Press, 2006.

[2] M. Assran et al. *Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture.* CVPR, 2023.

[3] V. Karpukhin et al. *Dense Passage Retrieval for Open-Domain Question Answering.* EMNLP, 2020.

[4] C. Burges et al. *Learning to Rank using Gradient Descent.* ICML, 2005.

[5] T. Schick et al. *Toolformer: Language Models Can Teach Themselves to Use Tools.* NeurIPS, 2023.

[6] S. G. Patil et al. *Gorilla: Large Language Model Connected with Massive APIs.* arXiv:2305.15334, 2023.