NEURIPS 2026

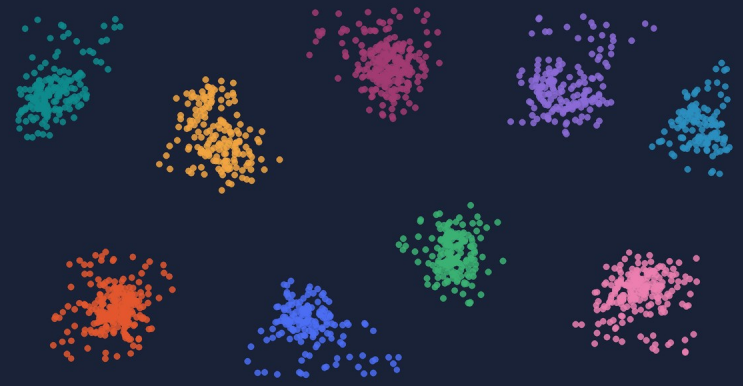
AutoManifold

Agentic Design of Data Visualisation Algorithms via Manifold Embedding

Burak Susam
University of Manchester

Ruiyuan Kang
X32 Studio

Tingting Mu
University of Manchester



PRESERVE
Balanced

MOTIVATION

Three algorithms, fixed trade-offs

t-SNE 2008

Minimises the mismatch between similarity distributions.

UMAP 2018

Aligns fuzzy topological structures.

PaCMAP 2021

Separates mid-near and far pairs to balance local and global fidelity.

Each encodes one immutable trade-off between local neighbourhood fidelity and global structure, fixed at design time.

MOTIVATION

Three algorithms, fixed trade-offs

t-SNE 2008

Minimises the mismatch between similarity distributions.

UMAP 2018

Aligns fuzzy topological structures.

PaCMAP 2021

Separates mid-near and far pairs to balance local and global fidelity.

Each encodes one immutable trade-off between local neighbourhood fidelity and global structure, fixed at design time.

The practitioner's only lever

Hyperparameter tuning over a narrow, algorithm-specific space.

The alternative

Hand-designing a new algorithm: specialised expertise and extensive trial and error.

THE QUESTION

Can LLM agents design the embedding algorithm itself, steered by the structure a user wants to preserve?

Naive code generation is insufficient:

- **Numerical errors**
- **Miscalibrated objectives**
- **Silent failures**
- **No link to user priorities**

METHOD · DESIGN SPACE

A unified embedding framework as skeleton

1 Difficulty analysis

DESIGNED difficulty assignment rule

FIXED patch clustering, scores

$$\mathrm{diff}(\mathbf{x}_i) \;=\; \mathrm{assign}\left(\sum_{l=1}^{L} \mathrm{score}_l(\mathrm{patch}(\mathbf{x}_i))\right) \;\in\; \{\mathrm{H},\, \mathrm{M},\, \mathrm{E}\}$$

METHOD · DESIGN SPACE

A unified embedding framework as skeleton

1 Difficulty analysis

DESIGNED difficulty assignment rule

FIXED patch clustering, scores

$$\mathrm{diff}(\mathbf{x}_i) \;=\; \mathrm{assign}\left(\sum_{l=1}^{L} \mathrm{score}_l(\mathrm{patch}(\mathbf{x}_i))\right) \;\in\; \{\mathrm{H},\, \mathrm{M},\, \mathrm{E}\}$$

2 Affinity ensemble

DESIGNED weights and feature maps

FIXED base affinity functions

$$a_{ij} \;=\; \sum_{l=1}^{Q} w_l\big(\mathbf{x}_i,\, \mathbf{f}^{(A)}(\mathbf{x}_i),\, \mathrm{diff}(\mathbf{x}_i),\, \mathbf{g}\big)\; a_l(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathrm{OptA}_l)$$

METHOD · DESIGN SPACE

A unified embedding framework as skeleton

1 Difficulty analysis

DESIGNED difficulty assignment rule

FIXED patch clustering, scores

$$\mathrm{diff}(\mathbf{x}_i) \;=\; \mathrm{assign}\left(\sum_{l=1}^{L} \mathrm{score}_l(\mathrm{patch}(\mathbf{x}_i))\right) \;\in\; \{\mathrm{H},\, \mathrm{M},\, \mathrm{E}\}$$

2 Affinity ensemble

DESIGNED weights and feature maps

FIXED base affinity functions

$$a_{ij} \;=\; \sum_{l=1}^{Q} w_l\big(\mathbf{x}_i,\, \mathbf{f}^{(A)}(\mathbf{x}_i),\, \mathrm{diff}(\mathbf{x}_i),\, \mathbf{g}\big)\; a_l(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathrm{OptA}_l)$$

3 Optimisation formulation

DESIGNED weights, objective selection

FIXED objective templates, solvers

$$\min_{\{\mathbf{z}_i\}} \sum_{l=1}^{H} \sum_{i=1}^{n} \lambda_l\big(\mathbf{x}_i,\, \mathbf{f}^{(O)}(\mathbf{x}_i),\, \mathrm{diff}(\mathbf{x}_i),\, \mathbf{g}\big)\; O_l(\mathbf{x}_i,\, \mathbf{z}_i,\, \mathrm{OptC}_l)$$

METHOD · DESIGN SPACE

A unified embedding framework as skeleton

1 Difficulty analysis

DESIGNED difficulty assignment rule

FIXED patch clustering, scores

$$\mathrm{diff}(\mathbf{x}_i) \;=\; \mathrm{assign}\left(\sum_{l=1}^{L} \mathrm{score}_l(\mathrm{patch}(\mathbf{x}_i))\right) \;\in\; \{\mathrm{H},\, \mathrm{M},\, \mathrm{E}\}$$

2 Affinity ensemble

DESIGNED weights and feature maps

FIXED base affinity functions

$$a_{ij} \;=\; \sum_{l=1}^{Q} w_l\big(\mathbf{x}_i,\, \mathbf{f}^{(A)}(\mathbf{x}_i),\, \mathrm{diff}(\mathbf{x}_i),\, \mathbf{g}\big)\; a_l(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathrm{OptA}_l)$$

3 Optimisation formulation

DESIGNED weights, objective selection

FIXED objective templates, solvers

$$\min_{\{\mathbf{z}_i\}} \sum_{l=1}^{H} \sum_{i=1}^{n} \lambda_l\big(\mathbf{x}_i,\, \mathbf{f}^{(O)}(\mathbf{x}_i),\, \mathrm{diff}(\mathbf{x}_i),\, \mathbf{g}\big)\; O_l(\mathbf{x}_i,\, \mathbf{z}_i,\, \mathrm{OptC}_l)$$

4 Configuration and refinement

DESIGNED configuration and update rules

FIXED reliability scores

$$\theta^{(t+1)} \;=\; F_{\mathrm{update}}\big(\theta^{(t)},\; \{\mathrm{score}_l\}_{l=1}^{L}\big)$$

METHOD · DESIGN SPACE

A primitive vocabulary to compose from

Affinity matrix — 6 PRIMITIVES

- `euclidean_knn`
- `mutual_knn`
- `snn`
- `geodesic`
- `lof_weighted`
- `learnable`

Objective function — 5 PRIMITIVES

- `kl_divergence`
- `cross_entropy`
- `spectral`
- `force_directed`
- `mutual_information`

Optimisation solver — 4 PRIMITIVES

- `sgd`
- `adam`
- `spectral_eigen`
- `fft_accelerated`

`hybrid affinity` `adaptive weighting` `multi-scale cost` `constrained multi-objective` `iterative refinement`

Composition patterns combine the primitives. The vocabulary guides rather than constrains: agents may depart from it.

METHOD · PIPELINE

Six phases, twelve specialist agents

INPUT Raw high-dimensional data + user priority

PLANNER AGENT fixed phase graph · owns the engines · enforces convergence

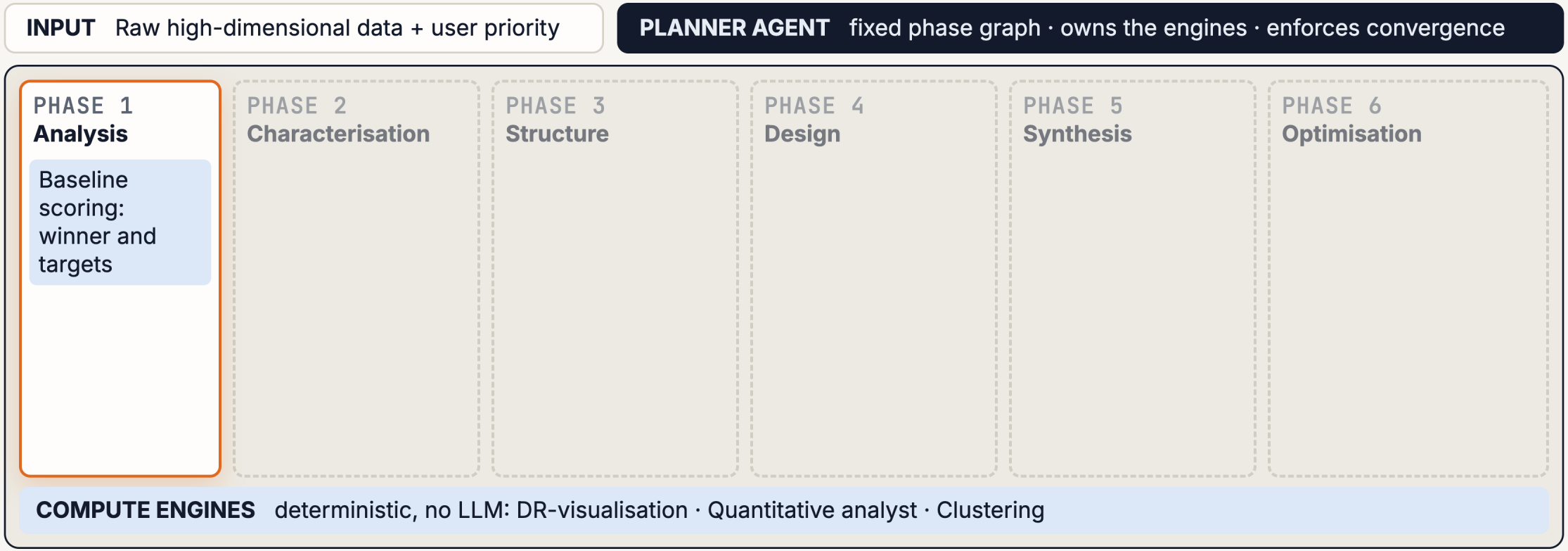
PHASE 1	PHASE 2	PHASE 3	PHASE 4	PHASE 5	PHASE 6
Analysis	**Characterisation**	**Structure**	**Design**	**Synthesis**	**Optimisation**

COMPUTE ENGINES deterministic, no LLM: DR-visualisation · Quantitative analyst · Clustering

Agents propose, engines measure. Each worker is a single-shot, schema-constrained LLM call. Deterministic engines compute every embedding and score, and the planner passes them on verbatim.

METHOD · PIPELINE

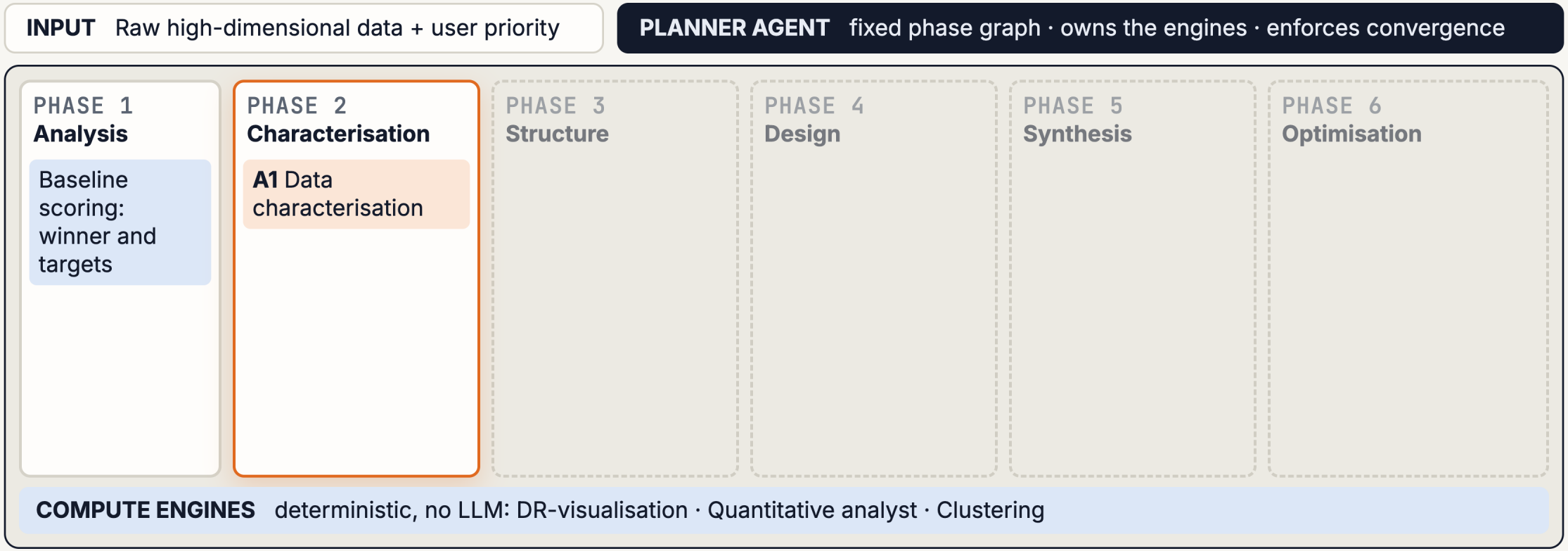
Six phases, twelve specialist agents



Phase 1 · Analysis. The engines run and score baseline embeddings such as PCA, t-SNE and UMAP. The best baseline sets the targets for the optimisation loop.

METHOD · PIPELINE

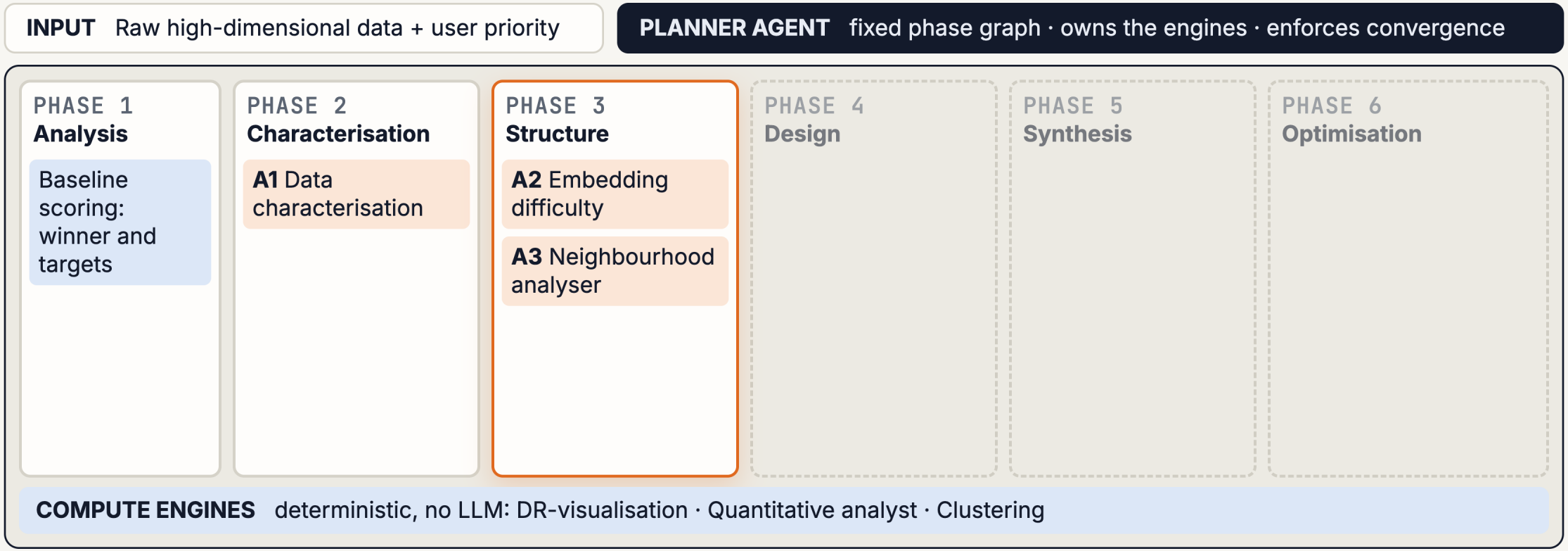
Six phases, twelve specialist agents



Phase 2 · Characterisation. Agent 1 designs the data characteristic function and the configuration initialisation, and recommends a pre-processing option.

METHOD · PIPELINE

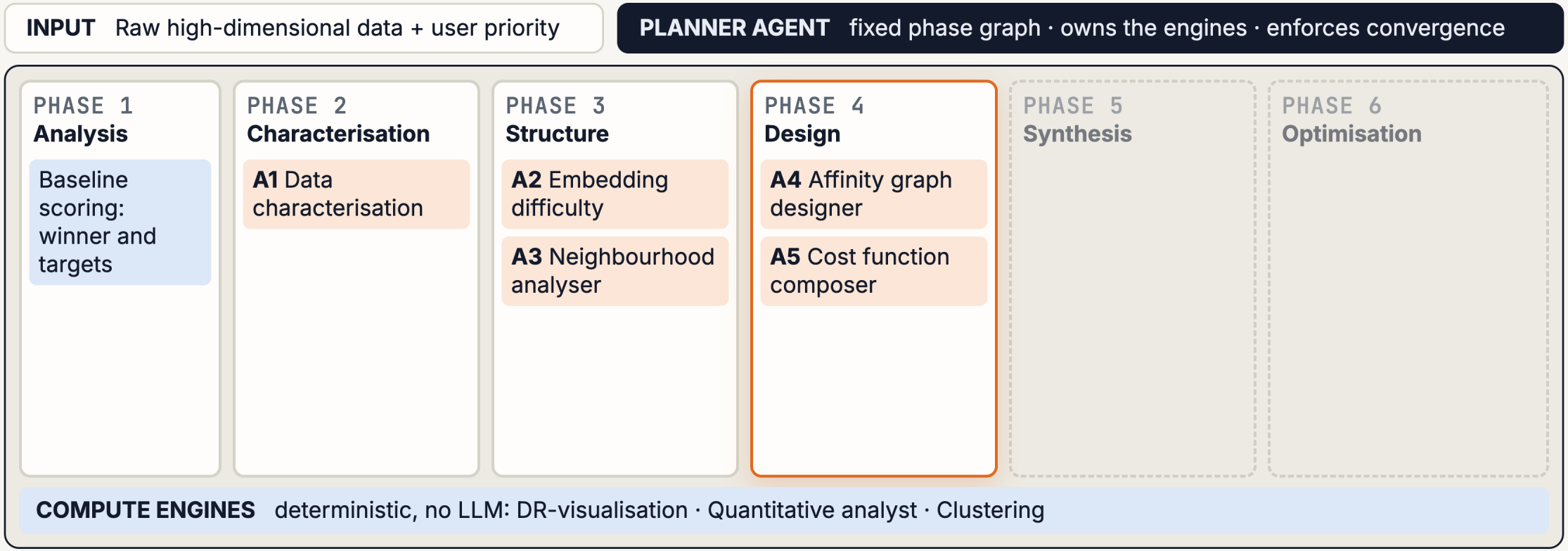
Six phases, twelve specialist agents



Phase 3 · Structure. Agent 2 designs the difficulty assignment; Agent 3 analyses neighbourhoods on the pre-processed data.

METHOD · PIPELINE

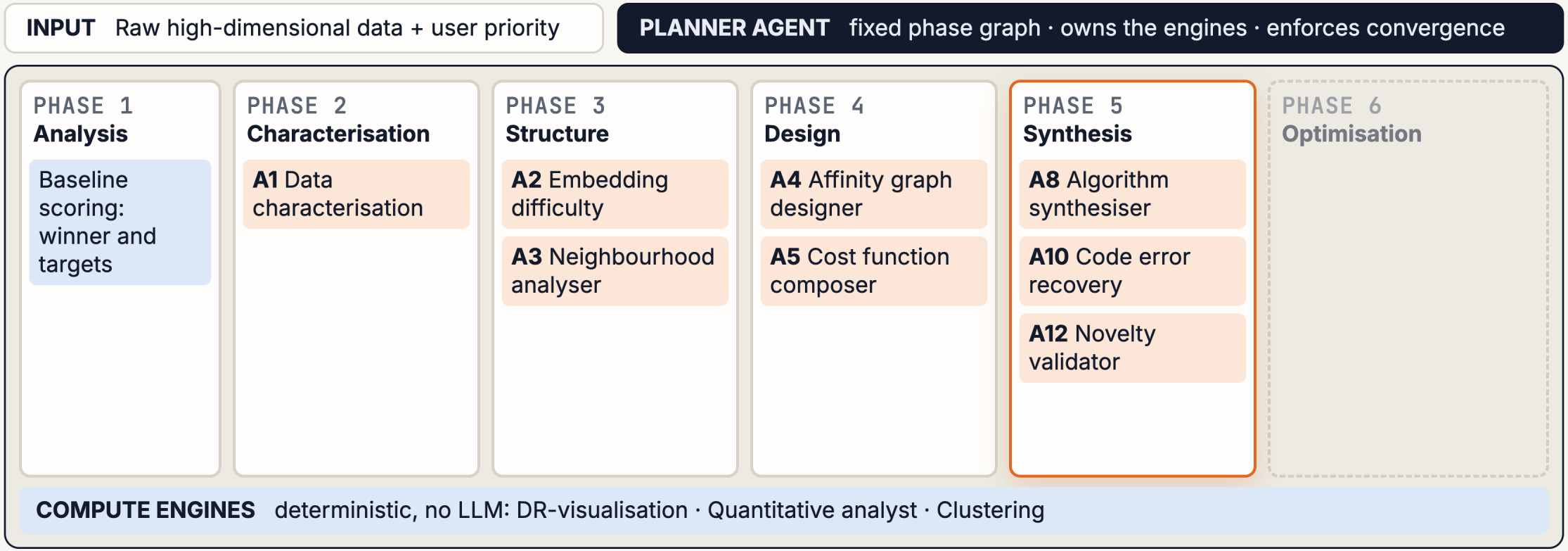
Six phases, twelve specialist agents



Phase 4 · Design. Agent 4 designs the affinity matrix and Agent 5 composes the optimisation formulation. The planner selects a solver.

METHOD · PIPELINE

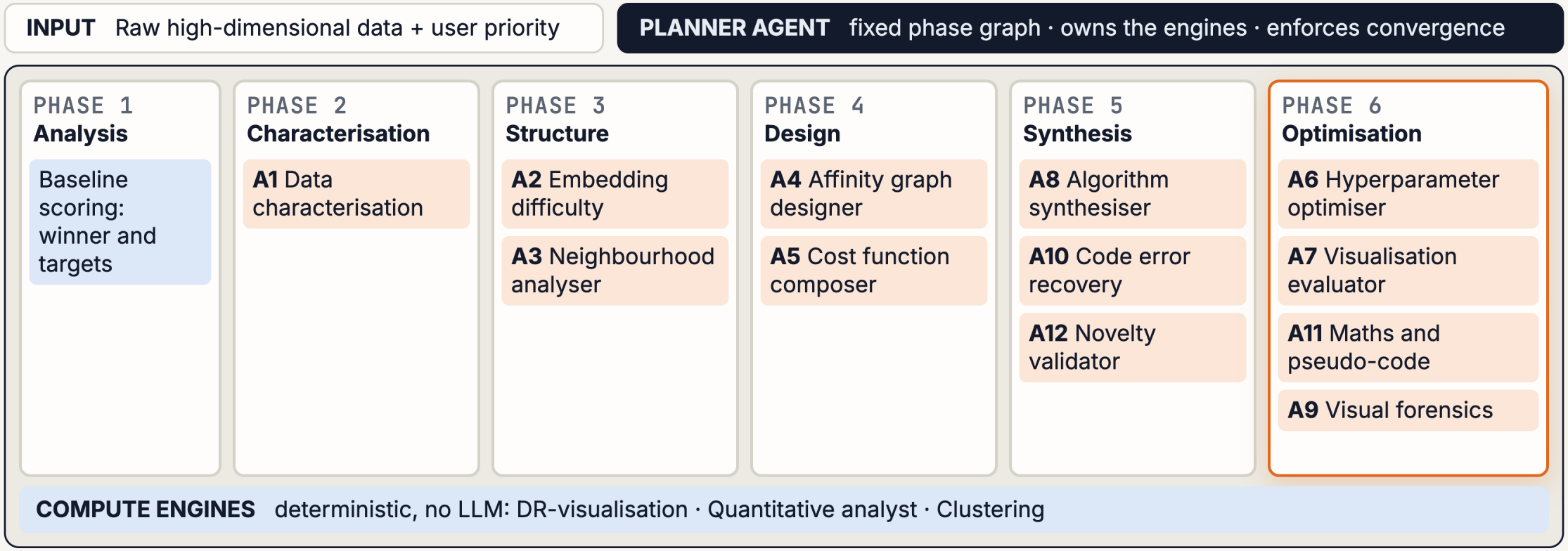
Six phases, twelve specialist agents



Phase 5 · Synthesis. Agent 8 writes the executable algorithm, Agent 10 recovers execution failures, and Agent 12 validates novelty against existing methods.

METHOD · PIPELINE

Six phases, twelve specialist agents



Phase 6 · Optimisation. Agent 6 refines hyperparameters on Agent 7's evaluation. Agent 11 writes the mathematics and pseudo-code; Agent 9 performs visual forensics.

METHOD · PIPELINE

Six phases, twelve specialist agents

INPUT Raw high-dimensional data + user priority

PLANNER AGENT fixed phase graph · owns the engines · enforces convergence

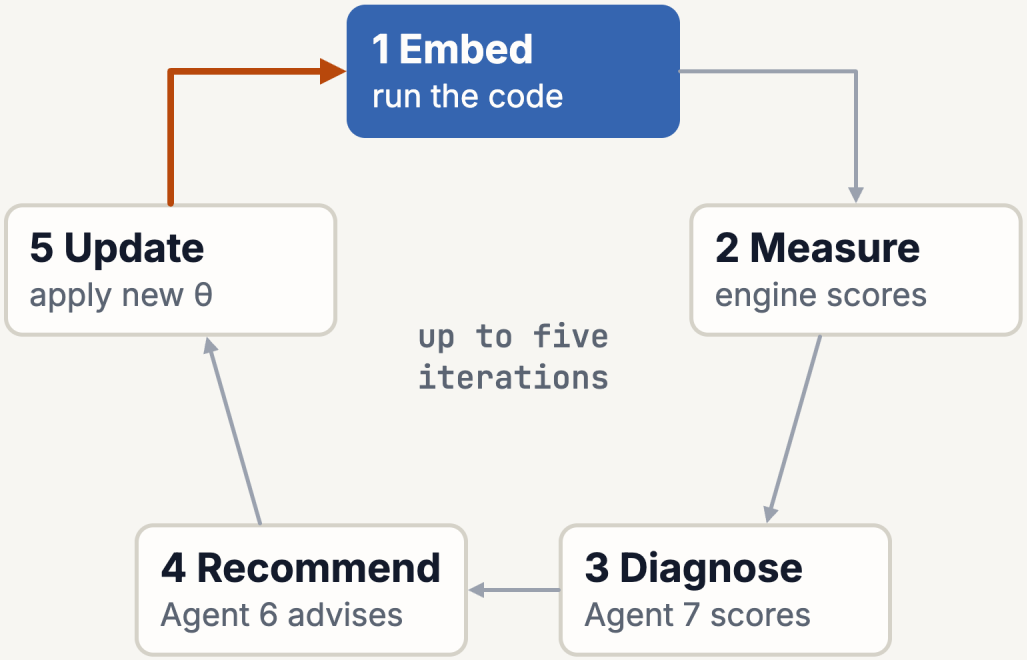
PHASE 1 **Analysis**	PHASE 2 **Characterisation**	PHASE 3 **Structure**	PHASE 4 **Design**	PHASE 5 **Synthesis**	PHASE 6 **Optimisation**
Baseline scoring: winner and targets	**A1** Data characterisation	**A2** Embedding difficulty	**A4** Affinity graph designer	**A8** Algorithm synthesiser	**A6** Hyperparameter optimiser
		A3 Neighbourhood analyser	**A5** Cost function composer	**A10** Code error recovery	**A7** Visualisation evaluator
				A12 Novelty validator	**A11** Maths and pseudo-code
					A9 Visual forensics

COMPUTE ENGINES deterministic, no LLM: DR-visualisation · Quantitative analyst · Clustering

OUTPUT Baseline comparison · evaluation metrics · mathematical formulation · pseudo-code · executable algorithm script

METHOD · REFINEMENT

A metric-grounded refinement loop



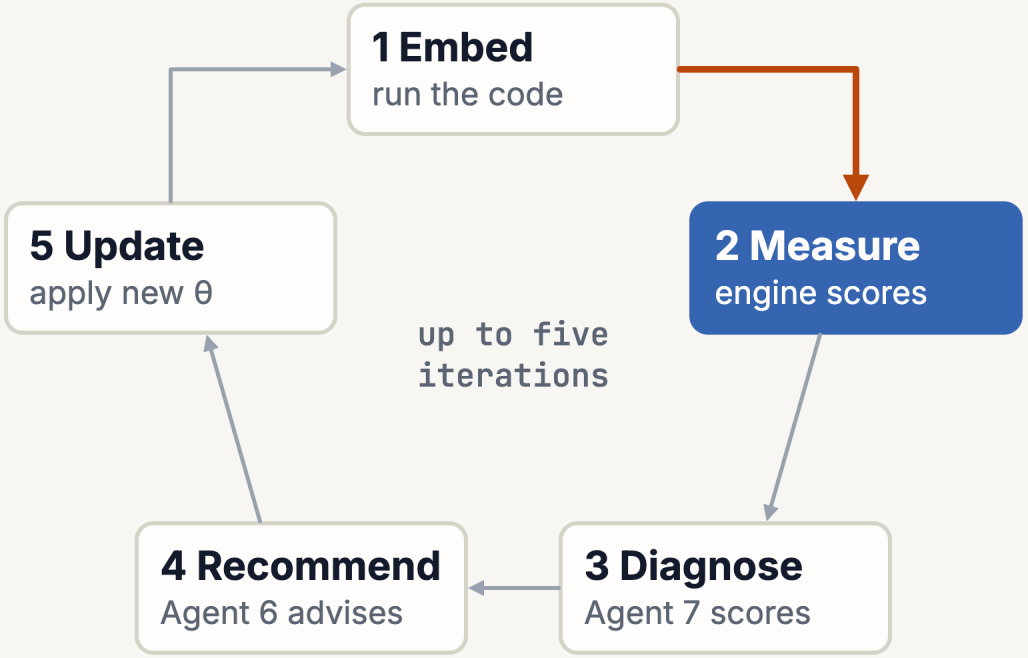
DETERMINISTIC ENGINE

Embed

The synthesised algorithm computes the embedding in an isolated namespace.

METHOD · REFINEMENT

A metric-grounded refinement loop



DETERMINISTIC ENGINE

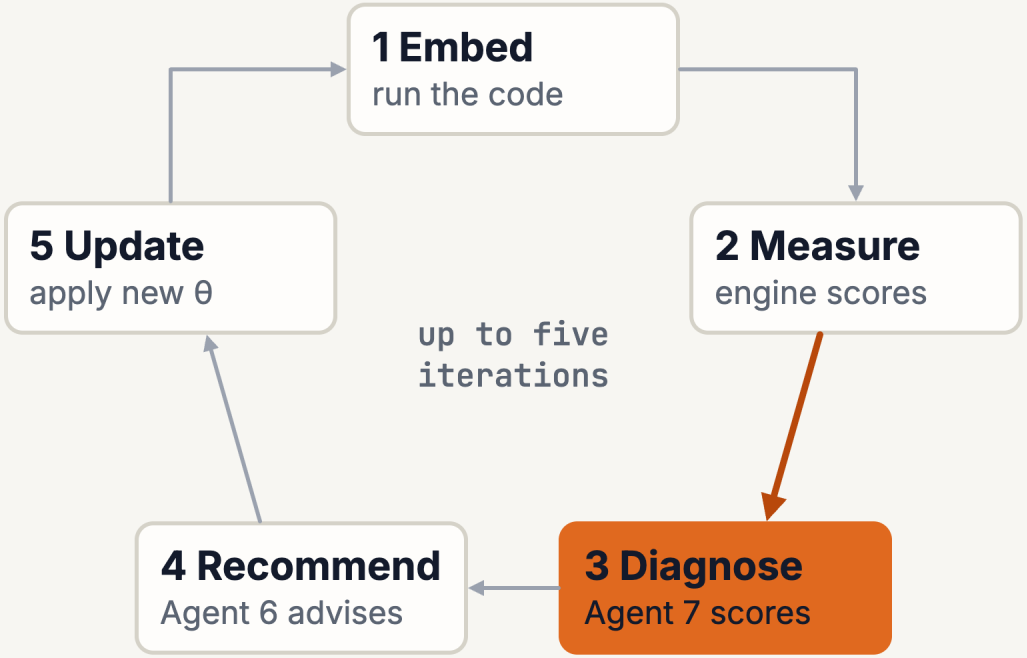
Measure

The quantitative analyst engine computes trustworthiness, continuity, Spearman correlation, stress, silhouette and LOF.

The values reach the agents verbatim.

METHOD · REFINEMENT

A metric-grounded refinement loop



LLM AGENT 7

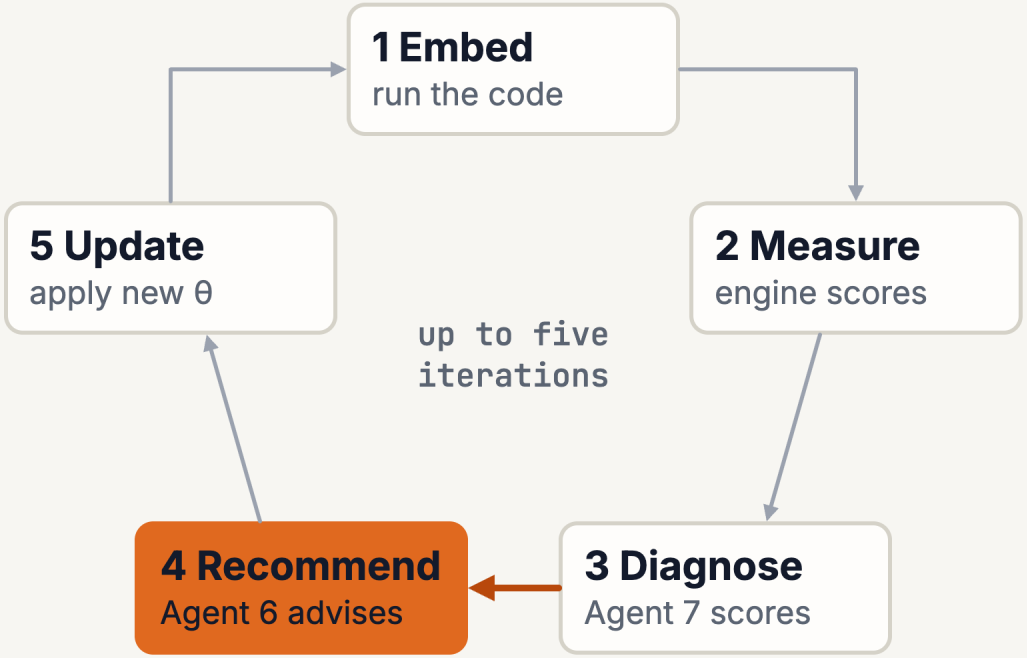
Diagnose

Agent 7 returns an explicit score: a composite of the metrics, weighted by the user's priority.

It is purely diagnostic and may not propose changes.

METHOD · REFINEMENT

A metric-grounded refinement loop



LLM AGENT 6

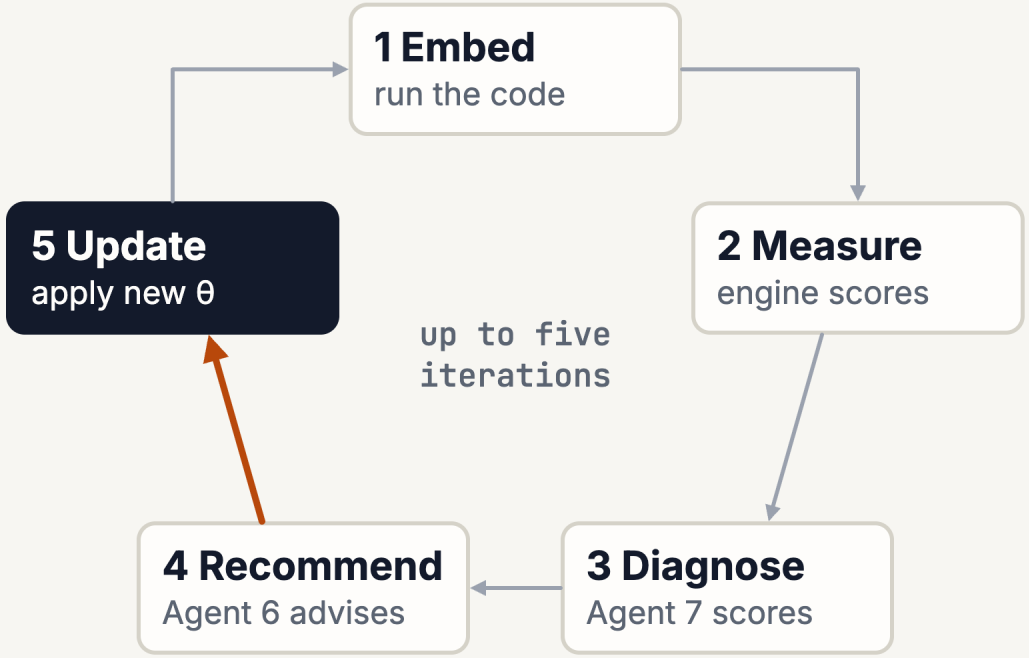
Recommend

Agent 6 adds an implicit score, a holistic judgement from 0 to 10, and recommends new hyperparameters.

It sees the full history: score changes, tried combinations, stale iterations.

METHOD · REFINEMENT

A metric-grounded refinement loop



PLANNER

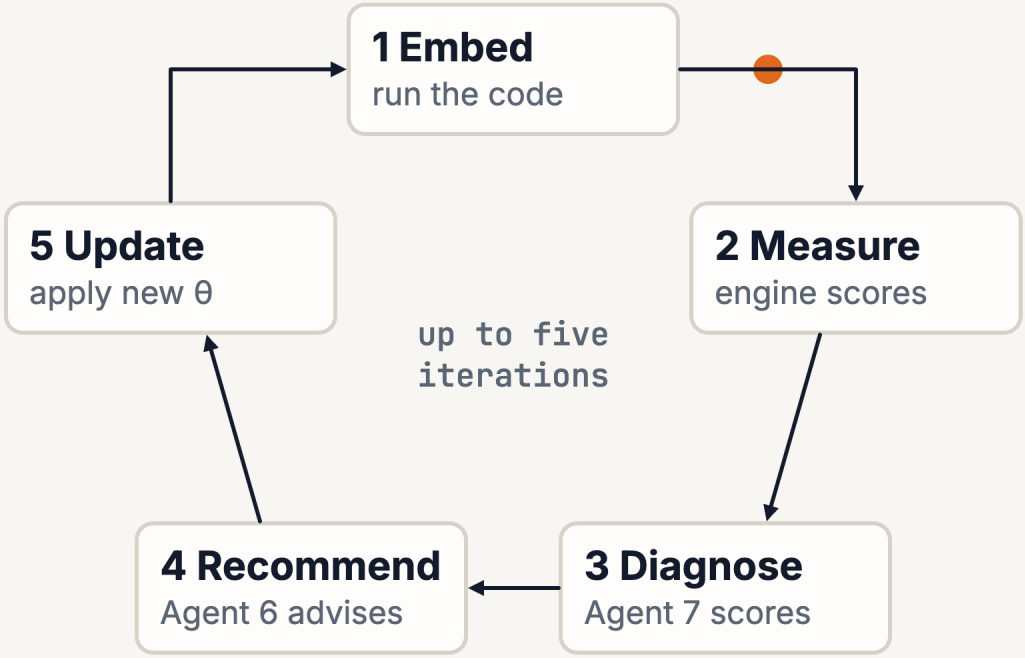
Update

The planner applies the recommendations and checks convergence.

$$\theta^{(t+1)} = F_{\text{update}}\left(\theta^{(t)},\ \{\text{score}_l\}_{l=1}^{L}\right)$$

METHOD · REFINEMENT

A metric-grounded refinement loop



STOPPING RULES

Repeat until

≥ 9	the score reaches 9 out of 10
2 ×	two consecutive iterations without improvement
5	iterations at most

Stale iterations are flagged; Agent 6 must then try bolder changes.

EXPERIMENTS

Evaluation under identical infrastructure

6

datasets

Easy: iris. Moderate: aging, kidney, pbmc3k. Hard: cortex, fashion-mnist.

3

baselines

t-SNE, UMAP and PaCMAP, tuned by random search.

6 + 4

reliability scores

Six drive the design. Four are held out: Shepard, NMI, RNX, kNN.

8

user priorities

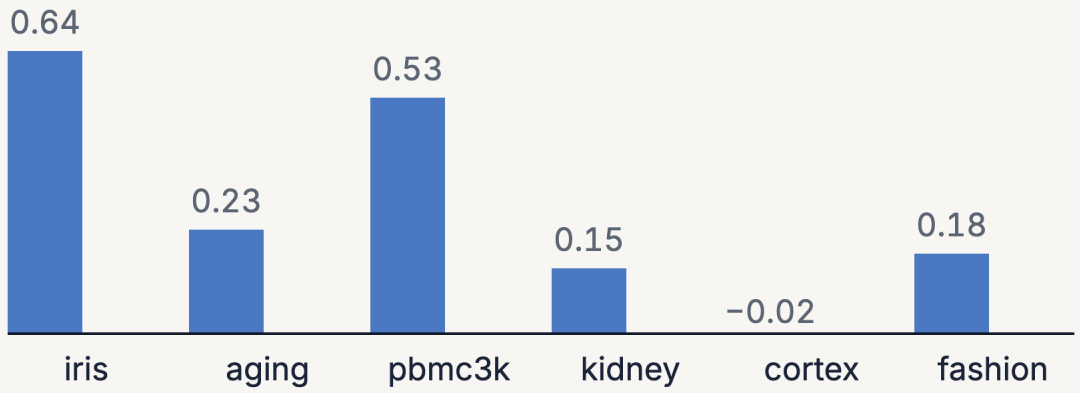
Balanced, local, global, or one of T, C, ρ, Silhouette, LOF.

All agents use Gemini-3.1-Pro. Alternative backbones are compared in the appendix.

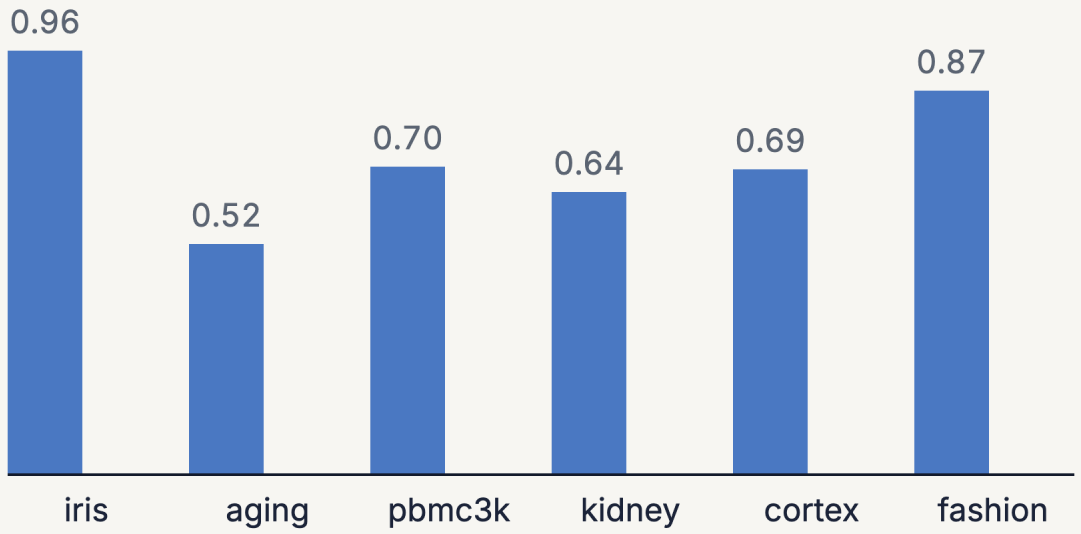
RESULTS

A priority steers the designed algorithm

Cluster separation: Silhouette ↑



Distance preservation: Spearman ρ ↑

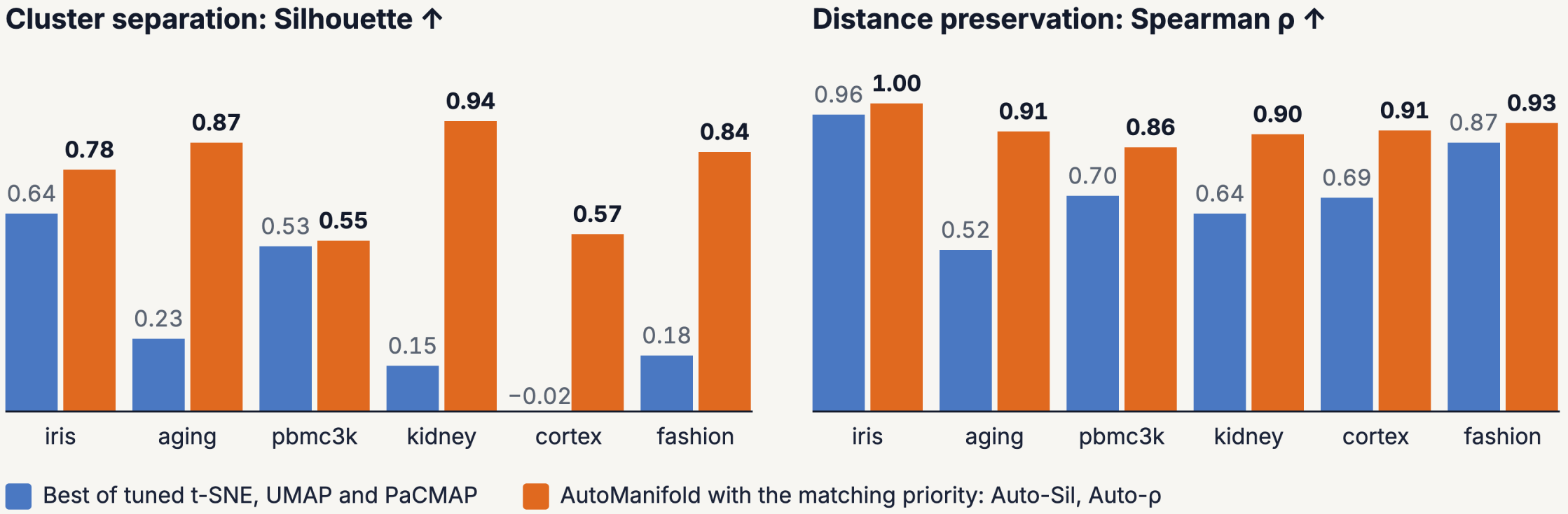


Best of tuned t-SNE, UMAP and PaCMAP

First, the strongest of the three tuned baselines on each dataset.

RESULTS

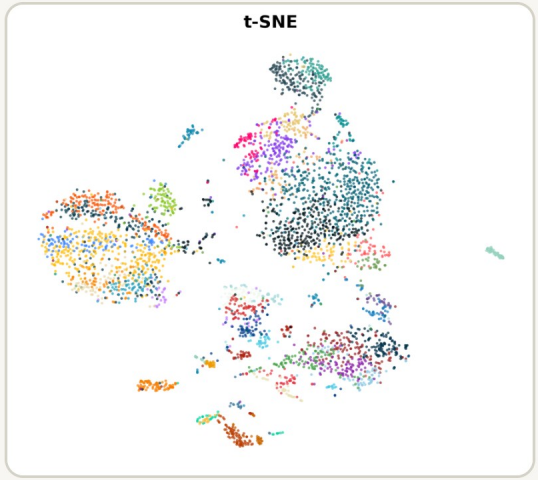
A priority steers the designed algorithm

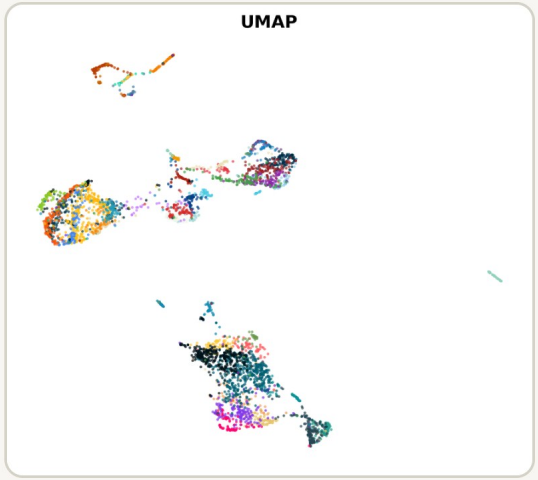


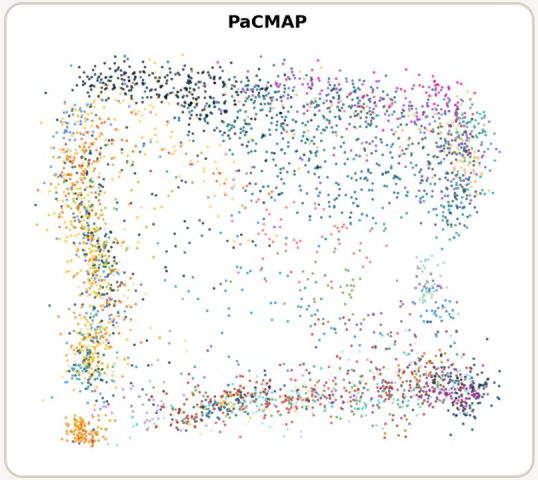
The prioritised metric exceeds the best baseline on all six datasets.

RESULTS

Cortex: 47 classes, one priority







Auto-Sil

priority: maximum Silhouette

t-SNE

Silhouette −0.015
Trustworthiness 0.973

UMAP

Silhouette −0.023
Trustworthiness 0.963

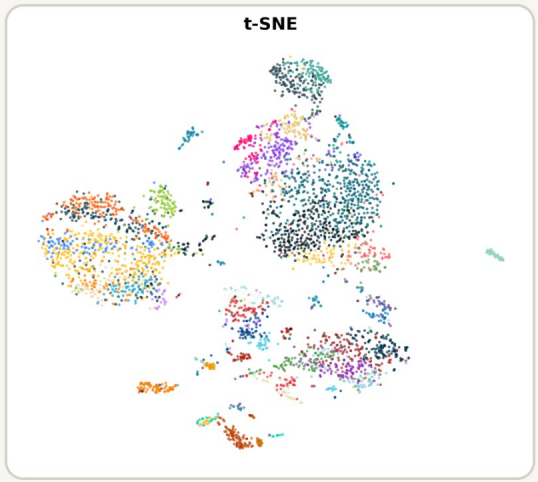
PaCMAP

Silhouette −0.020
Trustworthiness 0.966

Local scores above 0.96, yet the 47 classes overlap: Silhouette is below zero for all three.

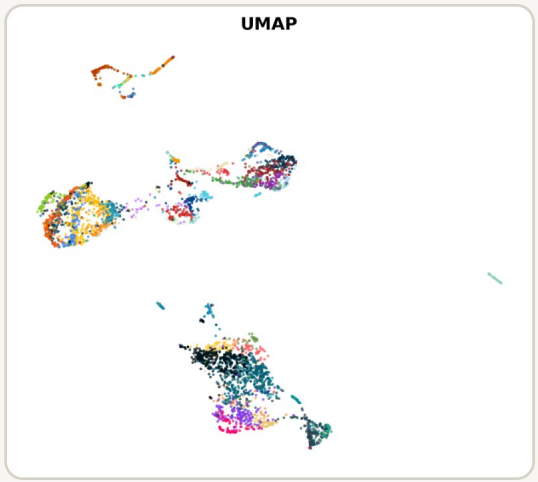
RESULTS

Cortex: 47 classes, one priority



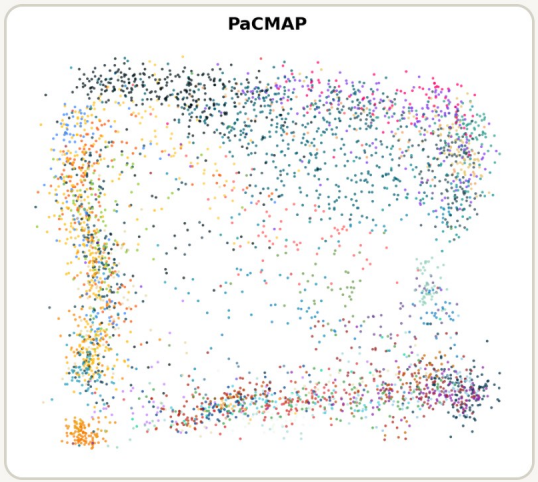
t-SNE

Silhouette −0.015
Trustworthiness 0.973



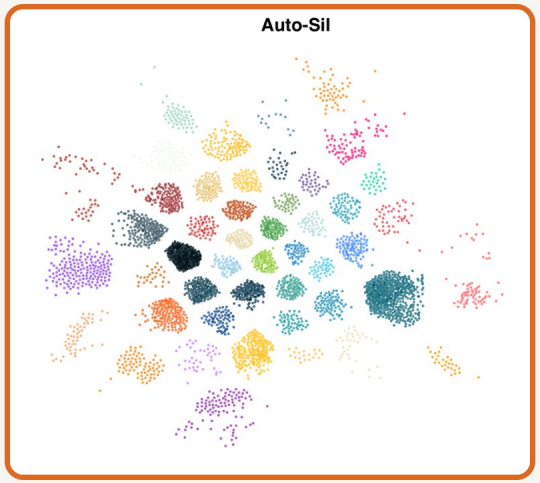
UMAP

Silhouette −0.023
Trustworthiness 0.963



PaCMAP

Silhouette −0.020
Trustworthiness 0.966



Auto-Sil

Silhouette 0.572
Trustworthiness 0.805

The same cells form compact, disjoint groups. The cost is local fidelity: trustworthiness 0.81.

RESULTS · GENERATED ALGORITHM

What the agents designed: DAS-SNE

Difficulty-Adaptive Silhouette t-SNE, synthesised for cortex under the maximum-Silhouette priority.

HYBRID AFFINITY

$$P_{\text{hyb}} = (I - W)\,P_{\text{gauss}} + W\text{SNN}$$

Gaussian affinities blended with shared-nearest-neighbour similarity. W is 0.15, 0.05 or 0, set by embedding difficulty.

OBJECTIVE

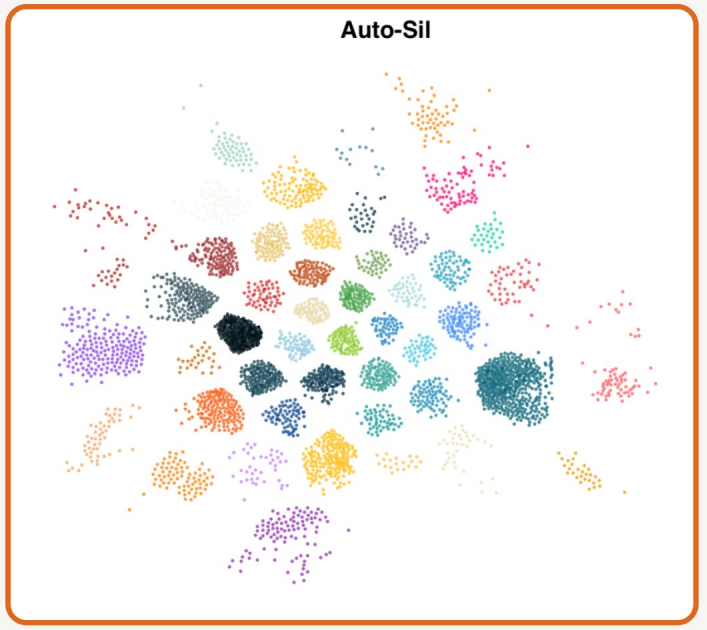
$$\min_{Y} \ \text{KL}\big(\tilde{P}(X),\, Q(Y)\big) + \lambda\, S(Y)$$

The t-SNE objective plus a silhouette-inspired term for compact, separated clusters.

UPDATE RULE

$$g = \nabla_{\text{KL}} Y + \lambda \frac{\|\nabla_{\text{KL}} Y\|}{\|\nabla_S Y\|} \nabla_S Y$$

The silhouette gradient is rescaled to the norm of the KL gradient.
Solver: Adam.



Silhouette 0.572 on cortex. Selected at iteration 4 of 5.

`AGENT 12: NOVEL VARIANT OF t-SNE`

DISCUSSION

Limitations and next steps

1 **Skeleton-based design space**
A fixed primitive library ensures numerical stability, but limits design freedom.

2 **Global priorities gain most**
Local metrics leave little headroom over t-SNE and UMAP, and steering one property costs others.

3 **One-shot algorithm**
The loop refines hyperparameters only; the functional form of the algorithm is not revised.

Next: systematic algorithmic revision inside the loop, and priorities that users adjust interactively.

AUTOMANIFOLD · NEURIPS 2026

A user priority goes in. A new embedding algorithm comes out.

With its mathematical formulation, pseudo-code and executable code, measured by deterministic engines at every step.

github.com/buraksusam/AutoManifold_NeurIPS2026
burak.susam@postgrad.manchester.ac.uk

Code and prompts