TAILORING THE CLUSTERING OBJECTIVE TO THE DATA: UTILITY-GUIDED VALIDITY-INDEX SELECTION AND CANDIDATE RERANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Choosing a clustering configuration without labels requires a criterion, and no single cluster-validity index (CVI) is reliable across geometries. Automated clustering systems adapt the criterion to the dataset, but coarsely: one predicted index, a fixed set, or one global quality surrogate. We present CLUSTOPT, which composes the criterion per dataset for two-dimensional data. Label-free meta-features are mapped to a predicted *utility profile*: for each of 60 indices, how well it would *rank* candidate partitions of the data at hand. The search objective is a weighted combination of the indices predicted to be informative, and a learned reranker makes the final choice among the candidates the search visited. To give the profile something to select among, we contribute 46 structural indices. On 1,055 controlled datasets held out from all learned components, weighting the vocabulary by the predicted profile is worth $+0.227$ ARI over weighting it uniformly. The profile also beats every alternative criterion we tested, including random search, the best single index, a static three-index core and a dataset-independent profile over the same vocabulary. The comparison shows *why*: all criteria visit candidates of near-identical quality, and differ by up to 0.26 ARI in which candidate they return. Selection is what matters, and CLUSTOPT improves it twice. Reranking the slate adds a further $+0.058$ ARI and recovers 61% of the distance to the best candidate visited, at under 1% of the search cost. On an independent 50-dataset benchmark CLUSTOPT has the highest mean of any system, a significant paired advantage over ML2DAC, and a lead over AutoClust on non-convex, connectivity-defined structure. Three generalist indices transfer across benchmarks; the remaining new indices are specialists whose value appears only under selection.

1 INTRODUCTION

Evaluating the quality of a clustering is one of the central problems of the field (von Luxburg et al., 2012). Every clustering algorithm has an objective — k-means minimizes within-cluster variance, a mixture model maximizes a likelihood — but those objectives are not comparable *across* algorithms. Run k-means, DBSCAN and spectral clustering on the same data and you obtain three partitions and no supervision with which to prefer one. In practice the gap is filled by a cluster-validity index (CVI): silhouette (Rousseeuw, 1987), Calinski–Harabasz (Caliński & Harabasz, 1974) or one of several dozen alternatives.

A CVI is sound only when it is informative about the structure present. Silhouette rewards compact, well-separated groups; density-based indices such as DBCV (Moulavi et al., 2014) respond to density connectivity; and the two disagree on exactly the geometries where the choice matters (Figure 1A). Benchmark studies have long shown that index quality varies with the data (Vendramin et al., 2010; Arbelaitz et al., 2013; Simpson et al., 2026b), and since the structure is unknown before clustering, committing to an index commits to an assumption about the answer. Automated clustering systems respond by adapting the criterion to the dataset (Section 2), but the adapted object is coarse: a single index, an unweighted set, or one surrogate shared by all datasets.

We ask a finer question of each index: on *this* data, would ranking candidate partitions by this index agree with ranking them by true quality? That quantity — the index's *utility* — is computable offline

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

wherever labels exist and predictable online from label-free descriptors. CLUSTOPT predicts a utility profile over a vocabulary of 60 indices, composes a search objective from the indices predicted to be informative, searches under it, and lets a learned reranker choose among the candidates the search visited. We study two-dimensional data with up to five clusters — spatial point patterns, interleaved signal trains, embeddings — where cluster *shape* carries the signal.

Contributions.

1. **Criterion adaptation at profile granularity.** We predict a continuous per-index utility profile from label-free meta-features and compose the search objective from it. Weighting the vocabulary by that profile rather than uniformly is worth $+0.227$ ARI; the profile beats every alternative criterion we tested, and conditioning on the dataset adds $+0.021$ $[+0.013, +0.030]$ over a dataset-independent profile, leading in 8 of 12 structural families (Section 5.1).
2. **Slate-aware candidate reranking.** A learned scorer over the visited candidates adds $+0.058$ ARI on the controlled benchmark and $+0.083$ on the independent one, at under 1% of the search cost (Sections 5.2, 7).
3. **A structural index vocabulary and its anatomy.** We contribute 46 structural indices and a 17,068-dataset generator with structural ground truth, and characterize which indices are generalists and which specialists (Section 6, Appendices B, W).

2 RELATED WORK

Automated clustering. Automated clustering inherits the combined algorithm-selection and hyperparameter-optimization formulation of supervised AutoML without its validation loss (Poulakis et al., 2024). Most systems adapt the criterion (Table 3, Appendix A): Muravyov & Filchenkov (2017) and ML2DAC (Treder-Tschechlov et al., 2023) predict one index, cSmartML (ElShawi et al., 2021) a set, AutoCluster (Liu et al., 2021) votes over per-index searches, AutoClust (Poulakis et al., 2020) learns one global surrogate, and AutoML4Clust (Tschechlov et al., 2021) leaves the index to the user. Others select the algorithm from meta-features or graph embeddings (Ferrari & de Castro, 2015; Pimentel & de Carvalho, 2019; Cohen-Shapira & Rokach, 2021), or a full pipeline (Singh et al., 2024; ElShawi & Sakr, 2022), in the algorithm-selection tradition (Rice, 1976; Kerschke et al., 2019). CLUSTOPT differs in the *granularity and semantics* of what is learned: a continuous per-index profile of how well ranking by that index would agree with ranking by true quality on this dataset, used to weight one scalar objective and recomputed per view.

Validity indices and their evaluation. Our utility target follows Vendramin et al. (2010), who score an index by the agreement between its ranking of candidate partitions and an external ranking; large comparative studies (Arbelaitz et al., 2013; Gagolewski et al., 2021; Simpson et al., 2026a) find no index that is best everywhere. Closest to our premise, Simpson et al. (2026b) map how the performance of nine indices varies with dataset features over 18,351 synthetic datasets; they analyze that variation, whereas we predict it per dataset and search under the prediction. Index ensembles (Jaskowiak et al., 2016) combine criteria with fixed rules; our uniform-weighting arm is the unselective limit of that idea. Our reranker is pointwise learning-to-rank over a generated slate (Liu, 2009); deep clustering (Xie et al., 2016; Caron et al., 2018) changes the representation and is complementary. Appendix U covers the index vocabulary (Davies & Bouldin, 1979; Halkidi & Vazirgiannis, 2001; 2008; Liu et al., 2013; Hu & Zhong, 2019; Gauss et al., 2023).

3 CLUSTOPT

Setting. Let D be a two-dimensional dataset and v a *view* of it: one of the two single-coordinate projections (x, y) or the joint view (xy). Single coordinates matter because some structure — interleaved pulse trains, parallel bands — separates along one axis and not in the plane. A configuration c (algorithm and hyperparameters) induces a partition π_c of (D, v); a validity index m_j maps a partition to a score, $j = 1, \ldots, M$. Automated clustering returns $\hat{c} = \arg\max_c J(\pi_c)$ for some label-free criterion J and is evaluated by $\mathrm{ARI}(\pi_{\hat{c}}, y)$ (Hubert & Arabie, 1985), which is unavailable at selection time. Choosing one index as J presupposes the kind of structure D contains. Prior systems relax this by predicting *which* index to use or by fixing a set in advance (Section 2); CLUSTOPT instead predicts, for every index, *how useful* it will be on (D, v) and builds J from that prediction, so the criterion is continuous, dataset-specific and recomputed per view. Appendix C specifies every component.

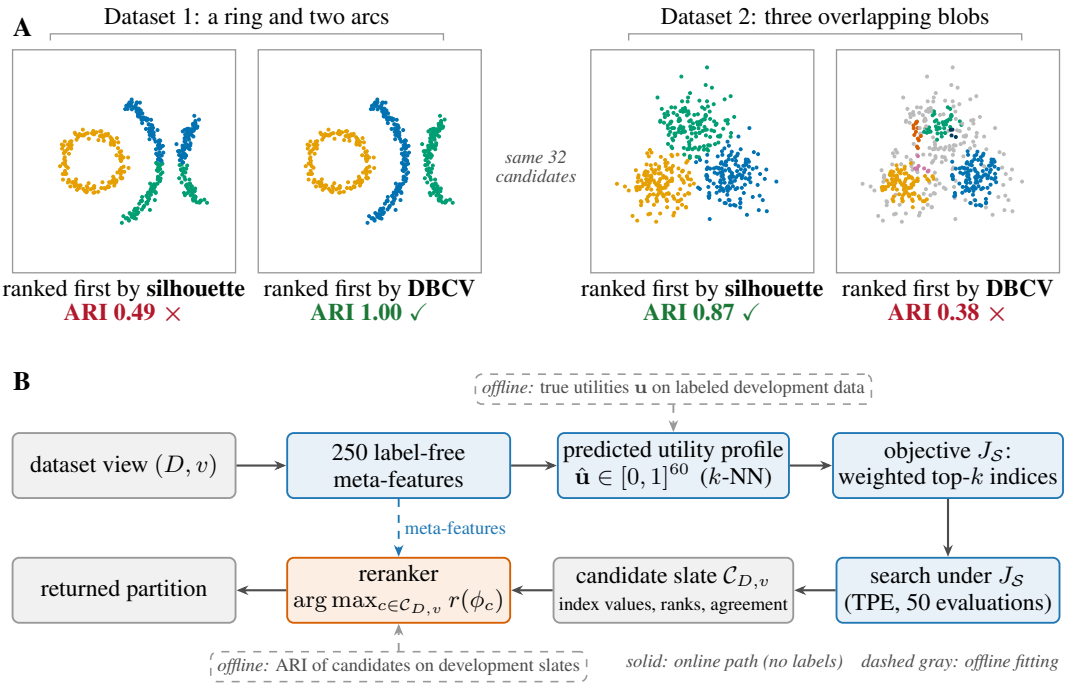

Figure 1: The problem and the method. *(A)* Two illustrative synthetic datasets, each with the partition that silhouette (compactness) and the partition that DBCV (density connectivity) rank first among the same 32 candidates, with the ARI of each: each index is right on one dataset and wrong on the other. *(B)* CLUSTOPT: meta-features give a predicted utility profile $\hat{\mathbf{u}}$, whose top-k indices form the objective J_S; a reranker then chooses among the visited candidates. Labels enter only offline (dashed gray).

Utility targets. Utility is the quantity everything else is built on. For one index on one dataset view it answers a single question: if candidate partitions were ranked by this index, how closely would that ranking agree with ranking them by true quality? It is defined on rankings, not index values, because the search only ever *compares* candidates and because index scales are not comparable. Following Vendramin et al. (2010), for Q candidates on (D, v) with index scores $\mathbf{s}_j$ and ARIs $\mathbf{q}$,

$$u_j \;=\; 0.7\, G_j(\mathbf{s}_j, \mathbf{q}) \;+\; 0.3\, T_j(\mathbf{s}_j, \mathbf{q}), \tag{1}$$

where G_j is *global* agreement (a fixed convex combination of positive-part Spearman and Kendall correlation, pairwise ordering accuracy and NDCG over all candidates) and T_j is *tail* agreement (overlap and NDCG in the top and bottom deciles, the bottom terms using $1 - \text{ARI}$ so that correctly rejecting bad partitions counts). Correlations are clipped at zero: an index anti-correlated with quality would misdirect the search and is not a sign-flipped good index. The eight component weights were fixed a priori and never tuned (Appendix C). Computed offline over a labeled repository, u_j turns index selection into a supervised problem: each dataset view yields a target vector $\mathbf{u} \in [0, 1]^{60}$, which the predictor below learns to map from label-free descriptions of the view.

The index vocabulary. FULL60 comprises 4 classical indices, 10 established variants and 46 new structural indices; HEAD14 is the first two groups and serves as the small-vocabulary control. Classical indices ask whether clusters are compact and separated; the new indices ask whether each cluster *looks like a coherent structure*, so that ring-, band-, lattice- and chain-shaped clusters have a criterion that speaks for them. Twenty-two indices act on point coordinates. For example, `curvature_consistency` measures whether signed curvature along a cluster's principal ordering keeps one sign and a steady magnitude, `ribbon_thickness` rewards an elongated cluster of near-constant width, and `axial_symmetry` whether the cluster mirrors about its principal axis. Twenty-four indices act on a rasterized image of the partition. For example, `gridness_fft_acf` scores lattice periodicity from the raster's autocorrelation, and Hough-transform indices score line and arc evidence. All are label-free, oriented so that larger is better, and defined in Appendix W.

Predicting the profile. Each view is described by 250 label-free meta-features — distributional, geometric and image-derived statistics plus landmarkers from cheap reference clusterings — none

using labels or the true cluster count. A $K{=}15$ nearest-neighbor regressor with inverse-distance weights maps them to $\hat{\mathbf{u}} \in [0,1]^{60}$ (Appendix C). On the 3,165 held-out Split-1 dataset–view records the deployed k-NN profile reaches mean Spearman 0.713 with the true profile, pairwise ranking accuracy 0.776 and NDCG 0.965 (neural: 0.762, 0.800, 0.971; Appendix E).

From a profile to an objective. The profile becomes a weighted objective over the subset $\mathcal{S}$ of the k indices with highest predicted utility,

$$J_{\mathcal{S}}(\pi) = \sum_{j \in \mathcal{S}} w_j \, \tilde{m}_j(\pi), \qquad w_j = \hat{u}_j \Big/ \sum_{l \in \mathcal{S}} \hat{u}_l, \tag{2}$$

with $\tilde{m}_j$ oriented so larger is better and normalized across the candidates seen so far. Keeping $\mathcal{S}$ small is deliberate: averaging over indices the profile ranks low reintroduces the indiscriminate use that Section 5.1 shows to be harmful. We use $k{=}10$, chosen on the development splits. A learned per-view selector over ten predictor×subset-size alternatives did not improve measurably on this fixed choice and costs 30% more runtime, so we do not deploy it (Appendix H).

Search. The space spans centroid-based, hierarchical, density-based, mixture and spectral algorithms, with $k \in \{2, \ldots, 5\}$ wherever a cluster count is needed; the true count is never supplied. Two regimes appear and are never mixed: a *fixed bank* of 32 configurations, which holds the candidate set constant so that only the objective varies, and *online search*, a Tree-structured Parzen sampler run under $J_{\mathcal{S}}$ for 50 evaluations, used for every end-to-end result (Appendix D).

Candidate reranking. The natural endpoint is to return the $J_{\mathcal{S}}$-maximizer, but $J_{\mathcal{S}}$ is built from a *predicted* profile and its maximizer is frequently not the best partition the search has visited. Let $\mathcal{C}_{D,v}$ be the *slate* of visited candidates. We learn a scorer $r(\phi(c, \mathcal{C}_{D,v}, D, v))$ and return its $\arg\max$ over the slate. The features ϕ comprise the candidate's own index values and partition descriptors, the view's meta-features, and *slate-context* features: each index's distribution over the slate, the candidate's rank under each index, and between-index agreement. The scorer is a gradient-boosted regressor of realized ARI — a pointwise ranker whose inputs are slate-relative — fitted with slate-level grouping on the development splits. It resembles AutoClust's CVI→ARI surrogate but is applied *after* search and sees the slate. It performs no clustering, so its cost is negligible, and the best candidate on the slate — the *slate oracle* — is its ceiling.

4 Experimental Design

We evaluate on two collections of datasets. A large synthetic repository with controlled, diverse structural traits supports large-scale evaluation and mechanism analysis; a smaller external corpus of benchmark and real datasets, frozen before any system ran, provides the independent test.

Controlled repository. 17,068 synthetic two-dimensional datasets in 12 structural families and 86 subfamilies, each family varying one geometric property, with 1,273–3,552 points and 2–5 clusters. Every dataset is drawn from a structural causal model over labeled components and stores a structural ground-truth record separate from its cluster labels (Appendix B). All learned components are fitted on splits 2–16; Split 1 (1,055 datasets) is used by none of them. Split 1 was consulted during development for system-level choices, so we treat it as a leakage-free instrument for mechanism analysis and rely on the external corpus for the independent test (Appendix A).

Independent external corpus. 50 datasets: 7 FCPS shape benchmarks, 15 scikit-learn generator datasets (five generators at three difficulty levels) and 28 real tabular datasets, 26 of them projected to two dimensions by PCA. It was frozen, with checksums, before any system ran, and its labels were read once after all executions finished. The 15 generator datasets are not independent, so all paired statistics resample the 40 *dependence groups* (Appendix O). PCA projections of tabular data are a weak test of any method (explained variance as low as 0.02; classes need not be clusters), which bounds what that half of the corpus can show.

Systems compared. We compare against two automated clustering systems and five criterion baselines:

- **AutoClust** (Poulakis et al., 2020): selects the algorithm by k-NN over meta-features and searches under a learned global CVI→ARI surrogate.

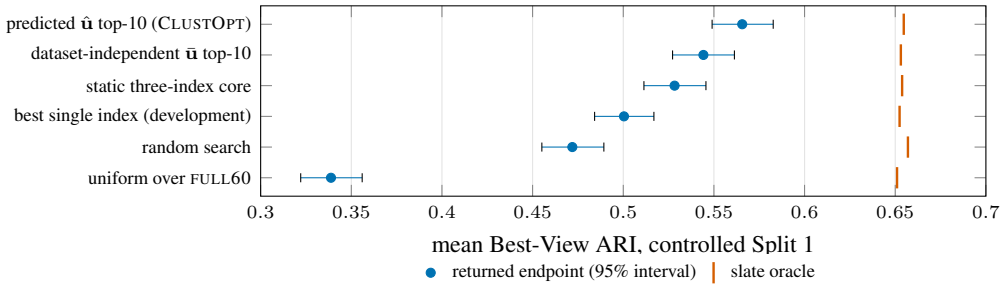

Figure 2: The criterion decides what is returned, not what is available. One row per search criterion on the 1,055 controlled Split-1 datasets (same space, sampler, budget and seeds). Dots: returned endpoints with 95% bootstrap intervals; orange ticks: mean slate oracles.

- **ML2DAC** (Treder-Tschechlov et al., 2023): predicts one index per dataset from meta-learned knowledge and warm-starts the search with it.
- **Random search**: 50 configurations with no criterion.
- **Uniform weighting**: equal weights over all of FULL60.
- **Best single index**: the index with the highest mean utility on the development splits, chosen per view.
- **Static core**: uniform weights over the three generalist indices of Section 6.
- **Dataset-independent profile** $\bar{\mathbf{u}}$: the mean true profile over the development splits, with the same vocabulary, k and weighting rule as CLUSTOPT — the direct ablation of conditioning.

Both systems are run with four index inventories (their original 7 classical indices, +10 established, +46 new, and all), with learned components fitted on the same splits 2–16 as ours (48,039 dataset–view records; Appendix Q). The criterion baselines share CLUSTOPT's search space, sampler, budget and seeds and differ only in J. All systems load the same prepared arrays and views. Candidate slates are shared only within an arm group (Table 12): within-group contrasts isolate final selection exactly; cross-system contrasts hold data, views, search domain and budget fixed, but each system follows its own search trajectory.

Metrics, statistics and cost. Quality is ARI against ground truth on all points, an algorithm's noise points forming one extra predicted cluster (internal indices are computed with noise removed); AMI (Vinh et al., 2010) gives the same conclusions where computable (Appendix I). We report *single-view* ARI on the xy view, which is what a user would deploy, and Best-View ARI, the maximum over the three views, which is an oracle over views and is always paired with the cost of running all three. The dataset is the inferential unit. Paired comparisons report the mean difference, a percentile bootstrap interval over dependence groups, a two-sided Wilcoxon signed-rank test (Wilcoxon, 1945), win/tie/loss counts and a rank-biserial effect size; pre-specified families are Holm-corrected (Holm, 1979). The criterion comparison of Section 5.1 is a seeded sweep, paired across arms; the reranking and system contrasts use an earlier replay of the deployed configuration; every external arm is a single seeded run (Appendix A). Reported runtimes cover search and scoring and exclude CLUSTOPT's meta-feature extraction (Appendix S).

5 CONTROLLED EVIDENCE

5.1 WEIGHTING, CONDITIONING, AND WHERE THE GAIN COMES FROM

This section evaluates the core of the method: the utility profile of Equation (1) as the basis of the criterion of Equation (2), independently of the baseline systems (Section 5.3). We ask whether weighting the vocabulary by the predicted profile beats using the same indices unweighted, whether it beats the criteria a practitioner would otherwise use, and how much of the gain comes from conditioning on the dataset rather than from knowing which indices are good on average.

Selective weighting makes the vocabulary useful. Crossing vocabulary (HEAD14, FULL60) with weighting (uniform; predicted profile; hindsight-best subset) under both search regimes (Appendix F) shows that how the indices are weighted matters far more than how many there are. Replacing

Table 1: Search criteria on the controlled Split 1 (1,055 datasets; online search, 50 evaluations per view; one seeded protocol). Every row differs only in the criterion J and returns that criterion's own maximizer, without reranking. *Endpoint*: mean Best-View ARI with 95% bootstrap interval; *Single*: the xy view; *Slate oracle*: the best candidate the search visited. Bold marks the best value in each block; paired contrasts and per-family results are in Appendices I and H.

Search criterion J	Endpoint	95% interval	Single	Slate oracle
Random search (no criterion)	0.472	[0.455, 0.489]	0.398	0.657
Uniform over FULL60	0.339	[0.322, 0.356]	0.306	0.651
Best single index (development)	0.500	[0.484, 0.517]	0.280	0.652
Static three-index core	0.528	[0.511, 0.546]	0.472	0.654
Dataset-indep. profile $\bar{\mathbf{u}}$, top-10	0.544	[0.527, 0.561]	0.422	0.653
Predicted profile $\hat{\mathbf{u}}$, top-10 (CLUSTOPT)	**0.566**	[0.549, 0.583]	**0.492**	0.655
CLUSTOPT, reranked (Section 5.2)[b]	**0.616**	[0.600, 0.633]	–	0.653
AutoClust, extended inventory[b]	0.614	–	–	–
ML2DAC, corrected, +established[b]	0.574	–	–	–

[b] Replay protocol, same datasets and search domain (Section 4); own trajectory per system.

uniform weights by the predicted profile is worth $+0.088$ ARI $[+0.079, +0.097]$ on HEAD14 with the candidate bank held fixed and $+0.239$ $[+0.225, +0.254]$ on FULL60 under online search; the deployed configuration gains $+0.227$ over the uniform arm of Table 1. The 46 new indices carry usable information — the hindsight-best subset of FULL60 beats that of HEAD14 by $+0.047$ — but only under selection: under uniform weights the same enlargement *costs* 0.058, whereas under the predicted profile it gains $+0.027$ (both significant; interaction $+0.085$). Enlarging a vocabulary without selecting within it dilutes the criterion.

The profile beats every alternative criterion, and conditioning adds to it. Uniform weighting is a weak opponent, since it includes indices that are anti-informative on a given dataset. Table 1 therefore compares the predicted profile with the criteria a practitioner would actually use, all under one seeded protocol. It leads random search by $+0.094$, the best single development index by $+0.065$, the static three-index core by $+0.037$ and the dataset-independent profile $\bar{\mathbf{u}}$ by $+0.021$; every interval excludes zero (Appendix I). The last contrast is the strict ablation of conditioning: $\bar{\mathbf{u}}$ uses the same vocabulary, the same k=10 and the same weighting rule, applied unchanged to every dataset. It is itself a strong criterion — knowing which indices are worth using on average accounts for most of the $+0.227$ — and predicting the profile from (D, v) adds to it consistently: $+0.021$ $[+0.013, +0.030]$, $p = 1.1 \times 10^{-10}$, 463 wins to 279 losses, and a lead in 8 of the 12 structural families, with six family-level intervals excluding zero and none in the opposite direction. On the deployable single view the margin is $+0.071$ $[+0.056, +0.085]$.

The gain is in selection, not in what the search visits. Figure 2 sets each criterion's returned endpoint against the best candidate its search visited. The criteria differ far less in what they make available than in what they return: mean slate oracles span 0.651 to 0.657 across all arms, a range of 0.006, against a range of 0.258 in the returned endpoints, and random search reaches the *highest* oracle of any arm and the second-worst endpoint. Decomposing each difference into a slate part and a within-slate selection part puts 93–102% on selection (Appendix I). Conditioning therefore works by choosing better from a comparable slate — which is also exactly the problem a second, learned selection step can address.

5.2 RERANKING THE SLATE THE SEARCH PRODUCED

Since every search visits a good partition and the criterion decides whether it is returned, the reranker of Section 3 adds a second, learned selection step; the question is how much of the remaining distance to the slate oracle it recovers. With the search held fixed, replacing the J_S-maximizer by the reranker's choice on the same slate is worth $+0.058$ ARI, interval $[+0.050, +0.066]$: it improves 530 of 1,055 datasets, leaves 320 unchanged and worsens 205, and recovers 61% of the distance to the slate oracle (0.653) with no additional clustering (Table 15). Across all 20 predictor×subset×weighting variants the gain lies between $+0.031$ and $+0.064$, every interval excluding zero, and it is largest where the criterion was weakest ($\rho = -0.96$; Appendix H). Because reranking, like conditioning, acts on which

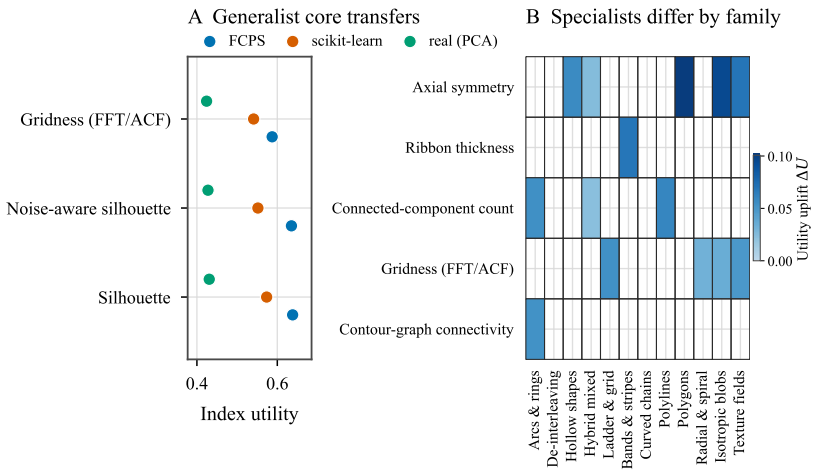

Figure 3: A generalist core and family specialists. *(A)* Utility of the three generalist indices on the independent corpus, by source: the ordering transfers, the level does not. *(B)* The 14 significant family-level specialist relations, colored by utility uplift ΔU inside the family against outside it.

candidate is returned, the two gains draw on the same headroom; whether a reranker trained for a non-conditioned criterion would close as much of it is a question we leave open (Appendix V.1).

5.3 Comparison with the automated clustering baselines

The lower block of Table 1 places the full CLUSTOPT pipeline beside AutoClust and ML2DAC on the same 1,055 datasets, configuration domain and budget. CLUSTOPT has the highest mean of the three. Against AutoClust with its extended inventory it is at parity: 0.616 against 0.614, paired difference $+0.002$ $[-0.004, +0.007]$, $p = 0.96$, 405 wins, 233 ties and 417 losses. Against ML2DAC with its corrected scorer on its best inventory it is clearly ahead: $+0.042$ $[+0.036, +0.049]$, $p = 1.0 \times 10^{-38}$, 642 wins to 282 losses. By structural family (Appendix K) the differences from AutoClust are small (six of 12 within ± 0.01), and the three whose intervals exclude zero point in the expected directions: CLUSTOPT ahead on arcs and rings ($+0.030$) and de-interleaving ($+0.013$), where clusters are extended and interleaved, AutoClust ahead on isotropic blobs (-0.031), where compactness is the right criterion. The external corpus tests whether either carries over.

6 Anatomy of the Vocabulary

What is there to select among? Figure 3 gives the answer: a small core of indices useful everywhere, and a larger set whose usefulness is confined to particular structural families. Recall that *utility* is how well an index orders candidate partitions by external quality. For every index and structural group we test whether the index is more useful *inside* the group than outside it (families against all other families, subfamilies against sibling subfamilies), and call the relation a *specialist relation* when the uplift is positive, its bootstrap interval excludes zero and it survives Benjamini–Hochberg correction. A *generalist* is an index in the top five by both true utility and selection frequency in at least six of the 12 families. Both definitions were fixed before computation (Appendix L).

A small core, and specialization below family level. Exactly three indices qualify as generalists, stably for thresholds of 5, 6 or 7 families: `silhouette` (classical), `noise_aware_silhouette` (established) and `gridness_fft_acf` (new) — one from each group (Figure 3A). Specialization lives mostly *below* family level: 14 family-level specialist relations survive correction (Figure 3B), spanning 10 of the 12 families; at subfamily level there are 42, of which 27 are invisible at family level. Specialist sets are nearly disjoint between families (mean pairwise Jaccard overlap 0.15, against 0.50 for top-5 utility sets): families agree on which indices are broadly good and disagree on which are specifically good, which is the structure a dataset-conditioned criterion can exploit and a shared one cannot. The specialists are the indices designed for the geometry in question: lattice and band descriptors on grids and stripes, symmetry and connected-component descriptors on rings and hollow

Table 2: Primary external corpus (50 datasets; mean ARI). *Single*: the deployable xy view; *Best-view*: an oracle over three views, broken down by source. RT: median three-view runtime in seconds, search and scoring only (Appendix S). Bold marks the best value per column; paired tests in the text support the claims. All 14 registered arms: Table 27.

			Best-view				
Code	System	Single	All	FCPS	sk-learn	Real	RT
C4	CLUSTOPT	0.319	**0.391**	**0.685**	0.753	0.123	48[a]
C1	without reranking	0.195	0.308	0.641	0.518	0.112	47[a]
C5	with policy selector	**0.328**	**0.399**	0.633	**0.774**	0.140	63[a]
A0	AutoClust, original (its best arm)	0.321	0.354	0.666	0.576	**0.157**	48
A3	AutoClust, extended	0.306	0.332	0.638	0.547	0.140	133
M1	ML2DAC, +establ. (its best arm)	0.223	0.324	0.526	0.633	0.108	20

[a] Excludes CLUSTOPT's meta-feature extraction, which is not measured (Appendix S).

shapes, ribbon and thickness descriptors on de-interleaving. The switch can be sharp: within the de-interleaving family, `gridness_fft_acf` is the specialist where the second coordinate drifts and `curvature_consistency` where the trains are straight and regular (Appendix N).

Does the system select what it should?. On the 3,165 Split-1 dataset–view records, the within-family correlation between an index's true utility and how often the deployed configuration selects it averages $\rho = 0.929$ (range 0.895–0.955; 0.891 on the new indices). The ten selected indices contain the true top-1 index on 72% of records and recover 65% of the true top-5 (55% on the new indices), so imperfect identification of the newer specialists plausibly accounts for part of the remaining predicted-to-oracle gap. (Appendix M).

7 INDEPENDENT EXTERNAL VALIDATION

The external corpus informed no design choice, and every arm on it was pre-registered.

End-to-end performance. Table 2 reports the deployed CLUSTOPT configuration, its two ablations, and the best and extended arm of each baseline. CLUSTOPT has the highest mean Best-View ARI of all systems (0.391, against 0.354 and 0.324 for the best AutoClust and ML2DAC arms) and the highest on two of three sources. Against ML2DAC the advantage is consistent under pairing: $+0.067$, interval $[+0.001, +0.157]$, $p = 0.039$, 30 wins to 15 losses, and it holds on every source ($+0.159$ on FCPS, $+0.120$ on the scikit-learn generators, $+0.015$ on the real tabular projections; Figure 4). Against AutoClust the mean difference is $+0.037$ but the interval $[-0.047, +0.134]$ spans zero ($p = 0.63$; 18 wins, 28 losses), and on the deployable single view the two are level (0.319 against 0.321): no overall advantage is established, and the difference is *structured*. By source it is $+0.178$ on the scikit-learn generators, $+0.019$ on FCPS and -0.034 on the real tabular projections, and within the generators it is carried by concentric circles and two moons (six datasets, two dependence groups, contributing $+0.067$): non-convex, connectivity-defined structures close to the *arcs and rings* family of our repository, where AutoClust's compactness-weighted surrogate returns the wrong partition (0.15 ARI on the easiest circles dataset, which DBSCAN separates perfectly) and a profile that promotes connectivity indices does not. On the real tabular projections every system's mean is below 0.16 (Appendix P.1). Adding the policy selector (C5) changes the mean by $+0.008$ with 40 of 50 datasets tied, at 30% more runtime.

Reranking, externally. The cleanest external ablation is the reranker: C4 and C1 share a criterion, a search and a slate and differ only in the returned candidate. Reranking is worth $+0.083$ ARI, interval $[+0.021, +0.159]$, $p = 0.0013$, rank-biserial 0.59, with 27 wins and 10 losses, at a median cost of 0.2 s against roughly 48 s of search (with the neural profile predictor, $+0.036$ with an interval spanning zero). Extending AutoClust's inventory to all 63 indices nearly triples its runtime and lowers its mean, the external counterpart of the uniform-weighting arm; CLUSTOPT's median cost matches AutoClust's (Appendix S).

Index utility transfers in ordering, not in level. Scoring 67 indices (the 60 deployed, three further classical ones and four recent comparators: CVDD, CVNN, DCSI, CDbw) on a separate 32-candidate

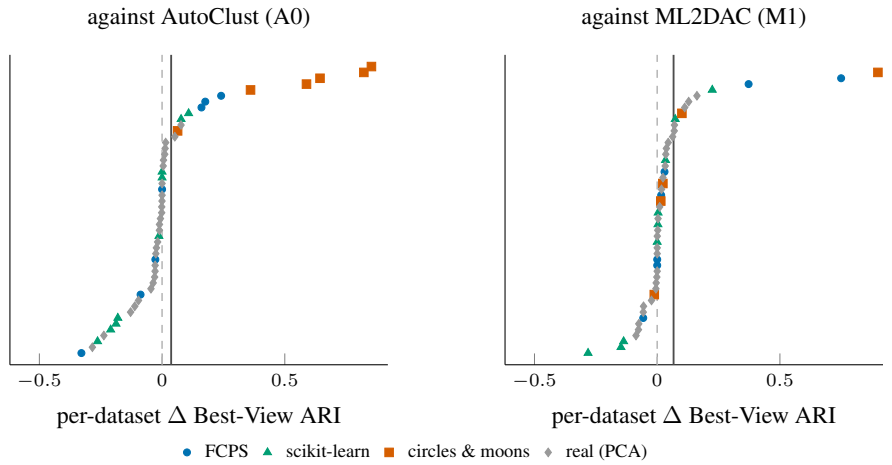

Figure 4: Where the external differences lie. Each point is one of the 50 datasets, sorted by the paired Best-View ARI difference between CLUSTOPT (C4) and the strongest arm of each baseline; the rule marks the mean. The advantage over ML2DAC is spread across sources; that over AutoClust is carried by the six concentric-circle and two-moon datasets (orange squares).

bank for each of the 150 external dataset–view units, the three generalists of Section 6 rank second, third and fifth overall (Table 29). Their ordering is stable across sources while their level is not. The 46 new indices have the lowest group mean yet several rank first on individual units: uniform use averages the specialists away; selection is what makes them worth having (Appendix R).

8 DISCUSSION AND CONCLUSION

A cluster-validity index is a statement about what structure the data contain, and our evidence is that the statement should be composed per dataset. A large vocabulary is a liability when used indiscriminately and an asset when weighted by a predicted utility profile: that weighting is worth $+0.227$ ARI over uniform use of the same indices, and the profile beats every alternative criterion we tested, including a dataset-independent one. Every criterion visits candidates of near-identical quality, so the difference between criteria lies in selection, and CLUSTOPT improves selection twice: through the conditioned criterion and through a reranker that recovers most of the remaining distance to the best candidate visited, at negligible cost, on both benchmarks. The vocabulary has the structure that makes this worthwhile: a small core that transfers, and specialists that are invisible in aggregate and decisive locally.

For a practitioner the upshot is that no validity index has to be chosen. Given a two-dimensional dataset, CLUSTOPT extracts label-free meta-features, composes its own criterion, searches under it and returns the reranked candidate, at the median cost of AutoClust and with no labels required at any point online; the vocabulary and the learned components are fixed once, offline, and reused across datasets.

Limitations. *Scope.* The method and all experiments concern two-dimensional data with at most five clusters; higher-dimensional inputs are reached only through projection. *Where it wins.* Externally, CLUSTOPT's advantage over AutoClust is concentrated in non-convex, connectivity-defined structure (Appendix P.1); on PCA projections of tabular data no system reaches a useful result, and a larger corpus of natively two-dimensional shape data would sharpen this comparison. *Open question.* Conditioning and reranking both act on selection; whether a reranker trained for a non-conditioned criterion would recover the same headroom is untested (Appendix V.1). *Protocol.* Best-View ARI is an oracle over three views, so single-view results are reported alongside; the external corpus has 40 dependence groups, each external arm is a single seeded run, and the controlled split informed design decisions. Predicting the view is the natural next step (Appendix V).

Reproducibility statement

We will release an anonymized repository (https://github.com/anonymousacount1/ClustOpt) containing the code, the dataset generator with all family and subfamily parameterizations, the configuration manifests, the index implementations and the derived result tables. Appendix A states the split construction, leakage control, statistical procedure, seeding and runtime accounting; Appendix B documents the generator; Appendix C specifies every learned component with its model class, hyperparameters, training population and seed; Appendix D gives the search space per view. Table 25 lists every external dataset with the identifier used to reconstruct it, its preparation parameters and the checksum of the prepared representation, and Table 26 gives the per-dataset results. We do not redistribute external raw or prepared data. Instead, the released loaders reconstruct each dataset from the listed source identifier and preparation protocol and verify the resulting prepared representation against the recorded checksum. Appendices F–S give the full numerical results behind every figure and table, and Appendix T records two corrections made to the external index-analysis layer.

Ethics statement

This work studies model selection for clustering on synthetic geometric data and public benchmark datasets. It involves no human subjects, no personally identifying data and no user study, and we are aware of no dual-use concern beyond those of automated data analysis generally.

AI use statement

Large language model (LLM) tools were used solely to assist with the writing of this manuscript, specifically for language polishing, including improving grammar, clarity, phrasing, and overall readability. These tools were not used to generate research ideas, design or conduct experiments, analyze results, or produce scientific content. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the content of this paper

References

Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013. doi: 10.1016/j.patcog.2012.07.021.

Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974. doi: 10.1080/03610927408827101.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Computer Vision – ECCV 2018*, volume 11218 of *Lecture Notes in Computer Science*, pp. 139–156, 2018. doi: 10.1007/978-3-030-01264-9_9.

Noy Cohen-Shapira and Lior Rokach. Automatic selection of clustering algorithms using supervised graph embedding. *Information Sciences*, 577:824–851, 2021. doi: 10.1016/j.ins.2021.08.028.

David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979. doi: 10.1109/TPAMI.1979.4766909.

Radwa ElShawi and Sherif Sakr. TPE-AutoClust: A tree-based Pipline ensemble framework for automated clustering. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1144–1153, 2022. doi: 10.1109/ICDMW58026.2022.00149.

Radwa ElShawi, Hudson Lekunze, and Sherif Sakr. cSmartML: A meta learning-based framework for automated selection and hyperparameter tuning for clustering. In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 1119–1126, 2021. doi: 10.1109/BigData52589.2021.9671542.

Daniel Gomes Ferrari and Leandro Nunes de Castro. Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences*, 301:181–194, 2015. doi: 10.1016/j.ins.2014.12.044.

Marek Gagolewski, Maciej Bartoszuk, and Anna Cena. Are cluster validity measures (in)valid? *Information Sciences*, 581:620–636, 2021. doi: 10.1016/j.ins.2021.10.004.

Jana Gauss, Fabian Scheipl, and Moritz Herrmann. DCSI – an improved measure of cluster separability based on separation and connectedness. *arXiv preprint arXiv:2310.12806*, 2023. doi: 10.48550/arXiv.2310.12806.

Maria Halkidi and Michalis Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. In *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 187–194, 2001. doi: 10.1109/ICDM.2001.989517.

Maria Halkidi and Michalis Vazirgiannis. A density-based cluster validity approach using multi-representatives. *Pattern Recognition Letters*, 29(6):773–786, 2008. doi: 10.1016/j.patrec.2007.12.011.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. doi: 10.2307/4615733.

Lianyu Hu and Caiming Zhong. An internal validity index based on density-involved distance. *IEEE Access*, 7:40038–40051, 2019. doi: 10.1109/ACCESS.2019.2906949.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075.

Pablo A. Jaskowiak, Davoud Moulavi, Antonio C. S. Furtado, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. On strategies for building effective ensembles of relative clustering validity criteria. *Knowledge and Information Systems*, 47(2):329–354, 2016. doi: 10.1007/s10115-015-0851-6.

Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. doi: 10.1162/evco_a_00242.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. doi: 10.1561/1500000016.

Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, and Sen Wu. Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, 43(3):982–994, 2013. doi: 10.1109/TSMCB.2012.2220543.

Yue Liu, Shuang Li, and Wenjie Tian. AutoCluster: Meta-learning based ensemble method for automated unsupervised clustering. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 12714 of *Lecture Notes in Computer Science*, pp. 246–258, 2021. doi: 10.1007/978-3-030-75768-7_20.

Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 839–847, 2014. doi: 10.1137/1.9781611973440.96.

Sergey Muravyov and Andrey Filchenkov. Meta-learning system for automated clustering. In *Proceedings of the International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms (AutoML@PKDD/ECML)*, volume 1998 of *CEUR Workshop Proceedings*, pp. 99–101, 2017.

Bruno Almeida Pimentel and André C. P. L. F. de Carvalho. A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences*, 477:203–219, 2019. doi: 10.1016/j.ins.2018.10.043.

Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. AutoClust: A framework for automated clustering based on cluster validity indices. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1220–1225, 2020. doi: 10.1109/ICDM50108.2020.00153.

Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. A survey on AutoML methods and systems for clustering. *ACM Transactions on Knowledge Discovery from Data*, 18(5):120:1–120:30, 2024. doi: 10.1145/3643564.

John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pp. 65–118. Elsevier, 1976. doi: 10.1016/S0065-2458(08)60520-3.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi: 10.1016/0377-0427(87)90125-7.

Connor Simpson, Ricardo J. G. B. Campello, and Elizabeth Stojanovski. Benchmarking of clustering validity measures revisited. *Statistical Analysis and Data Mining*, 19(1):e70061, 2026a. doi: 10.1002/sam.70061.

Connor Simpson, Mario Andrés Muñoz, and Ricardo J. G. B. Campello. Instance space of clustering validation measures. *Data Mining and Knowledge Discovery*, 40(3):25, 2026b. doi: 10.1007/s10618-026-01195-x.

Prabhant Singh, Pieter Gijsbers, Murat Onur Yildirim, Elif Ceren Gok, and Joaquin Vanschoren. CLAMS: A system for zero-shot model selection for clustering. *arXiv preprint arXiv:2407.11286*, 2024. doi: 10.48550/arXiv.2407.11286.

Dennis Treder-Tschechlov, Manuel Fritz, Holger Schwarz, and Bernhard Mitschang. ML2DAC: Meta-learning to democratize AutoML for clustering analysis. *Proceedings of the ACM on Management of Data*, 1(2):144:1–144:26, 2023. doi: 10.1145/3589289.

Dennis Tschechlov, Manuel Fritz, and Holger Schwarz. AutoML4Clust: Efficient AutoML for clustering analyses. In *Proceedings of the 24th International Conference on Extending Database Technology (EDBT)*, pp. 343–348. OpenProceedings.org, 2021. ISBN 978-3-89318-084-4.

Lucas Vendramin, Ricardo J. G. B. Campello, and Eduardo R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209–235, 2010. doi: 10.1002/sam.10080.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(95):2837–2854, 2010.

Ulrike von Luxburg, Robert C. Williamson, and Isabelle Guyon. Clustering: Science or art? In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pp. 65–79, 2012.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. doi: 10.2307/3001968.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 478–487, 2016.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008. doi: 10.1613/jair.2490.

A Reproducibility and evaluation protocol

Naming key. The appendices keep the identifiers under which artifacts are archived. In main-text terms: *Stage 1* is the vocabulary×weighting factorial of Section 5.1; *Stage 2* is the fitting of profile predictors, rerankers and policy selectors; *Stage 3* is the index analysis of Section 6; *Stage 4* is the primary external benchmark of Section 7. A *policy* is one predictor×subset-rule×weighting variant of Equation (2); the deployed CLUSTOPT is the policy `knn_top10_raw` with its reranker

Table 3: Criterion adaptation across automated clustering systems. Adapting the clustering criterion to the dataset is established prior work; CLUSTOPT differs in the granularity and semantics of the adapted object, not in whether one is adapted. "Granularity" is what the learned component emits per dataset.

System	Criterion adapted?	Learned component and granularity	Criterion applies to	Also adapted
AutoML4Clust	no	index supplied by the user	—	search budget, algorithm
Muravyov & Filchenkov (2017)	yes	meta-learned prediction of *one* validation index	configuration search	algorithm
cSmartML	yes	meta-learned selection of algorithm *and* evaluation criteria; a criterion *set* per dataset, whose members enter as separate objectives	evolutionary tuning against multiple objectives	algorithm, hyperparameters
AutoCluster	yes	meta-learned algorithm shortlist; tuned under *several* CVIs separately, partitions then combined by a *fixed* majority-voting rule	each grid search under its own CVI	algorithm, output ensemble
ML2DAC	yes	meta-learned choice of *one* CVI + warm start and a reduced search space	Bayesian optimisation	search region, algorithm
AutoClust	yes, but *globally*	k-NN algorithm selection over CVI-valued meta-features + one *global* network from a candidate's CVI vector to predicted ARI	Bayesian optimisation (TPE)	algorithm, hyperparameters
CLUSTOPT	yes	predicted *continuous utility profile* over the whole index vocabulary; objective composed from a weighted top-k of that profile, recomposed per dataset view	configuration search	slate reranking (learned separately)

(external arm C4). PP V1 and PP V2 are the two versions of the per-view policy selector that the main text mentions once and does not deploy (Appendix H). *Tier A* denotes what the main text calls a significant specialist relation. An *arm group* is a set of external arms that share one physical search trace. "Holm-rejected" means significant after Holm correction within the pre-specified family.

Table 4: Every index inventory used in the paper. Parenthesized totals are the inventories the historical baseline runs actually received: two image-based indices were missing from the baselines' index matrix because of a name-lookup defect, not because they are uncomputable (see text).

Inventory	Classical	Establ.	New	Recent	Total (realized)	Used by
HEAD14	4	10	–	–	14	CLUSTOPT, small-vocabulary control
FULL60	4	10	46	–	60	CLUSTOPT
Original	7	–	–	–	7	baselines, arm 0
+Established	7	10	–	–	17	baselines, arm 1
+New46	7	–	46	–	53 (51)	baselines, arm 2
Extended	7	10	46	–	63 (61)	baselines, arm 3
Index analysis	7	10	46	4	67	Sections 6, 7

HEAD14 and FULL60 contain four of the seven classical indices: silhouette, Calinski–Harabasz, Davies–Bouldin and DBCV. The other three — Dunn, COP and Coggins–Jain — are ML2DAC-side indices supplied by ML2DAC's own index collection; CLUSTOPT's metric registry does not implement them and never computes a utility target for them, so they cannot enter its utility vocabulary, and they are reported as not available rather than imputed. The omission reflects which system

implements an index, not its performance; its effect is that the baselines' Original inventory is strictly richer than CLUSTOPT's classical subset.

Two new indices, `convexity_ratio_image_based` and `hough_arc_circle_strength`, are live in CLUSTOPT: their utility targets are finite on all 48,039 development records, and the deployed configuration places them in its top ten on 121 and 137 of the 3,165 Split-1 records. Externally both are computable on every unit. In the historical baseline runs, however, their columns were empty because the baseline index-evaluation path looked values up by registry key while the evaluator returned them under each index's internal name, and the two differ only for these indices. The +New46 and Extended arms of both baselines were therefore run with 51 and 61 indices instead of the intended 53 and 63. We report those arms as the inventories they realized rather than the ones they requested; the numerical effect of the shortfall has not been re-evaluated, which would require rebuilding the index matrix and re-running the affected arms.

Splits and leakage control. The controlled repository is divided into 16 splits. Utility targets, meta-features, both utility predictors, the policy selectors and the ten rerankers are fitted on splits 2–16 with an inner validation carve-out grouped by dataset, so no dataset contributes to both training and validation. Recorded train–test and validation–test dataset overlaps are zero.

Split 1 (1,055 datasets) is excluded from all learned-component fitting, and it is *not* a pristine untouched holdout. Earlier analyses on Split 1 informed later system-level design decisions — which policy families to carry forward, and the decision to retrain the policy selector against reranker-aware targets. The final Split-1 evaluation reported in Section 5 is therefore a *controlled, development-informed replay*: leakage-free with respect to learned-component training, high-powered, and the right instrument for mechanistic analysis, but not an independent test of the finished design. The independent external benchmark of Section 7 is the independent test, and it was used once, end to end.

Views. Each dataset is evaluated under three views: the two single-coordinate projections (`x_only`, `y_only`) and the joint two-dimensional view (`xy_2d`). Meta-features are extracted per view, so the predictor may prefer different indices for different projections of the same data. Best-View ARI maximizes over views using ground truth and is an oracle in that respect; the three-view runtime is the cost of producing the candidates it selects among.

External corpus preparation. Each external dataset is cleaned, imputed where required, scaled and projected to two dimensions once. The resulting array is checksummed and archived, and every system loads that exact array; all 50 checksums match the frozen manifest. Preparation happens outside the measured runtime boundary and is identical across systems. The ground-truth boundary was opened once, after all executions completed.

Metrics. Quality is the adjusted Rand index against ground truth. Points that a density-based algorithm labels as noise (label -1) are neither dropped nor converted to singletons: ARI is computed on all points with the raw predicted labels, so all noise points together form one additional predicted cluster. The same convention is used for AMI and on the external corpus. Internal indices are computed differently: noise points are removed before scoring, except by `noise_aware_silhouette`, which scores the original labels, and a partition must retain at least two clusters after removal to be scored (Appendix D). Adjusted mutual information (Vinh et al., 2010) is reported for the headline contrasts in Appendix I.

Seeds and repetitions. Seeding per stage is specified in Appendix D. The controlled Split-1 end-to-end traces were produced by a single run on the unseeded path. Every external arm is a single seeded run, and we report it as such: no multi-seed average is claimed anywhere, and between-seed variation on the external corpus has not been measured. The controlled criterion comparison of Section 5.1 is a separate, fully seeded execution (`base_seed` 42, 15,825 runs) in which all arms share a dataset–view seed and are therefore paired by construction; the controlled Split-1 replay behind the reranker analysis predates that work and was not repeated with seeds. The reranker contrast is unaffected by this, because its two values come from the same run over the same visited slate.

Statistics. The dataset is the inferential unit. Bootstrap intervals are percentile intervals over 10,000 resamples that resample *dependence groups* rather than rows; the external corpus has 40 such groups over 50 datasets, because the five scikit-learn generators each contribute three difficulty variants. Wilcoxon signed-rank tests are two-sided; where a family of contrasts was pre-specified,

the Holm-corrected decision is reported and is what we treat as the outcome. Effect sizes are matched-pairs rank-biserial correlations. Family-level and subfamily-level quantities in Section 6 use Benjamini–Hochberg correction within their pre-specified test family, which is stated with each table.

What is shared between arms. All systems consume the same frozen prepared representations and the same view definitions. Physical candidate traces are shared only within predefined arm groups, listed in Table 12. Comparisons within a group isolate final selection exactly; comparisons across systems hold data, views and search domain fixed but not the realized candidate set, and are never described as sharing a candidate slate. The 32-configuration bank used for the index analysis of Section 6 is a separate metric-analysis track constructed independently of the end-to-end executions.

Runtime accounting. A system is charged the full physical search cost it consumed plus its own scoring cost. Where several variants reused one candidate slate, each is charged the full slate cost as though it had run alone; no variant receives a discount for sharing. Deduplicated wall-clock time for the whole external benchmark was $38{,}084$ s against $49{,}997$ s of charged method-equivalent time, and the difference is an implementation saving that is never passed to any arm. Only physical search and arm scoring carry non-zero values in the persisted records; meta-feature extraction and model initialization are logged as zero and are not inferred. Because CLUSTOPT computes 250 meta-features per view, including landmark clusterings and rasterized-image descriptors, that logging convention understates its cost relative to the baselines. We have not re-measured meta-feature extraction, for CLUSTOPT or for the baselines' own meta-features, so every runtime in this paper describes the measured execution boundary and nothing wider. In particular we do not claim that CLUSTOPT's *total* cost equals AutoClust's; what the records support is that their search-and-scoring costs are comparable and that the reranker adds 0.2 s to a 48 s three-view search.

Numerical environment. Scientific workers run with numeric thread counts pinned. The recorded environment is Python 3.12 with NumPy 2.1.3, SciPy 1.14.1 and scikit-learn 1.6.0. Corpus preparation was performed once under unpinned BLAS and archived, so it is never recomputed during scientific execution.

B THE CONTROLLED SYNTHETIC REPOSITORY

B.1 GENERATION MECHANISM

The repository is produced by a structural causal generator (internally `HYBRID_SCM_V2`) rather than by a monolithic sampler, and it is not a black box: every dataset carries the complete parameterization that produced it.

A dataset is a set of labeled *components*. Each component has a type drawn from a structural vocabulary — `ring_annulus`, arc segments, polyline and corner chains, parabolic chains, parallel bands, lattice and ladder elements, polygon frames, spiral and meander forms, texture and density fields, and isotropic Gaussian blobs — together with the type's own geometric parameters. For a ring annulus, for example, these are the center, the inner and outer radii, the angular extent, the ellipse ratio and the hole offset. Each component draws a requested number of points, receives its own additive noise (type and scale recorded per component), and carries its own cluster label. Components may declare *parents*, which is how hybrid and nested families are composed, and optional *operators* may be applied to a component after sampling.

Three properties make the repository usable as evidence rather than as decoration. First, every dataset is generated in the unit square $[0, 1]^2$, so scale is not a confound between families. Second, three seeds are recorded per component — for parameters, for points and for noise — alongside the dataset seed and the family seed, and each dataset directory stores a *replay configuration* sufficient to regenerate it exactly. Third, each dataset stores a *structural ground truth* record separate from its cluster labels: the true component count, hole count, junction and endpoint counts, symmetry axes, periodicity, parallel-group and ladder-rung counts, blob centers, ridge orientations, and the contour and skeleton graphs. That record is what makes the family-level claims in Section 6 interpretable: when an index is useful on a subfamily, the property it is responding to is recorded independently of any clustering.

B.2 SCALE, FAMILIES AND SUBFAMILIES

The corpus contains 17,068 datasets across 16 splits, evaluated under three views each, giving 51,204 dataset–view records; splits 2–16 supply 16,013 records per view for fitting and Split 1 supplies 1,055 datasets for the controlled replay. Every dataset has between 1,273 and 3,552 points, a generator cluster count in $\{2, 3, 4, 5\}$, and one of four declared difficulty levels (`easy`, `medium`, `hard`, `very_hard`). The true cluster count is recorded but never made available to any method.

Table 5: The controlled repository: 12 structural families over the whole 16-split corpus.

Family	Subfamilies	Datasets	Points (min–max)	Median points	k
Arcs & rings	8	1700	1422–3535	2562	2–5
Bands & stripes	7	1500	1469–3445	2580	2–5
Curved chains	7	1267	1706–3392	2582	2–5
De-interleaving	7	1733	1325–3494	2567	2–5
Hollow shapes	7	1400	1355–3493	2576	2–5
Hybrid mixed	8	2033	1599–3536	2585	2–5
Isotropic blobs	7	1000	1454–3509	2610	2–5
Ladder & grid	7	1334	1382–3448	2584	2–5
Polygons	7	1167	1440–3379	2584	2–5
Polylines	7	1400	1480–3552	2565	2–5
Radial & spiral	7	1267	1466–3432	2571	2–5
Texture fields	7	1267	1273–3435	2579	2–5
Total	86	17068	1273–3552		2–5

Table 6: All 86 subfamilies of the controlled repository.

Family	Subfamily	Datasets	Points	Difficulty levels
Arcs & rings	`annuli_variable_thickness`	200	1602–3293	easy/medium/hard/v.hard
Arcs & rings	`broken_rings`	220	1526–3323	easy/medium/hard/v.hard
Arcs & rings	`clean_arcs`	200	1557–3308	easy/medium/hard/v.hard
Arcs & rings	`full_rings`	200	1577–3375	easy/medium/hard/v.hard
Arcs & rings	`nested_arcs`	200	1662–3535	easy/medium/hard/v.hard
Arcs & rings	`nested_overlapping_arc_pairs`	200	1422–3477	easy/medium/hard/v.hard
Arcs & rings	`noisy_overlapping_arcs`	280	1660–3373	easy/medium/hard/v.hard
Arcs & rings	`partial_circles`	200	1505–3245	easy/medium/hard/v.hard
Bands & stripes	`clean_parallel_bands`	220	1631–3376	easy/medium/hard/v.hard
Bands & stripes	`crossing_or_near_crossing_bands`	220	1470–3284	easy/medium/hard/v.hard
Bands & stripes	`fragmented_bands`	220	1702–3311	easy/medium/hard/v.hard
Bands & stripes	`noisy_interleaved_bands`	240	1774–3445	easy/medium/hard/v.hard
Bands & stripes	`rotated_parallel_bands`	200	1469–3285	easy/medium/hard/v.hard
Bands & stripes	`thick_ribbons`	200	1759–3443	easy/medium/hard/v.hard
Bands & stripes	`variable_spacing_bands`	200	1728–3286	easy/medium/hard/v.hard
Curved chains	`arc_parabola_mixture`	180	1706–3329	easy/medium/hard/v.hard
Curved chains	`clean_parabolas`	180	1742–3283	easy/medium/hard/v.hard
Curved chains	`curves_with_density_drift`	177	1832–3392	easy/medium/hard/v.hard
Curved chains	`fragmented_curves`	180	1827–3384	easy/medium/hard/v.hard
Curved chains	`noisy_curved_ribbons`	190	1799–3217	easy/medium/hard/v.hard

Family	Subfamily	Datasets	Points	Difficulty levels
Curved chains	opposite_direction_parabolas	180	1751–3371	easy/medium/hard/v.hard
Curved chains	variable_curvature_chains	180	1749–3259	easy/medium/hard/v.hard
De-interleaving	amp_arc_trends	250	1325–3324	easy/medium/hard/v.hard
De-interleaving	hybrid_deinterleaving_scene	263	1607–3334	easy/medium/hard/v.hard
De-interleaving	missing_pulses	240	1650–3309	easy/medium/hard/v.hard
De-interleaving	pri_amp_coupled	260	1630–3344	easy/medium/hard/v.hard
De-interleaving	pri_parallel_trains	240	1753–3494	easy/medium/hard/v.hard
De-interleaving	pri_with_jitter	240	1358–3412	easy/medium/hard/v.hard
De-interleaving	spurious_pulses_noise	240	1648–3362	easy/medium/hard/v.hard
Hollow shapes	bridged_components	200	1486–3304	easy/medium/hard/v.hard
Hollow shapes	broken_hollow_shapes	200	1702–3339	easy/medium/hard/v.hard
Hollow shapes	disconnected_components	200	1545–3493	easy/medium/hard/v.hard
Hollow shapes	hollow_mixed_shapes	210	1355–3279	easy/medium/hard/v.hard
Hollow shapes	multi_hole_shapes	200	1803–3366	easy/medium/hard/v.hard
Hollow shapes	single_hole_rings	190	1620–3364	easy/medium/hard/v.hard
Hollow shapes	topological_noise	200	1606–3202	easy/medium/hard/v.hard
Hybrid mixed	arcs_plus_blobs	250	1634–3376	easy/medium/hard/v.hard
Hybrid mixed	bands_plus_arcs	270	1710–3340	easy/medium/hard/v.hard
Hybrid mixed	bands_plus_blobs	250	1666–3536	easy/medium/hard/v.hard
Hybrid mixed	curves_plus_ladders	260	1603–3298	easy/medium/hard/v.hard
Hybrid mixed	fragmented_band_with_overlapping_clouds	200	1800–3365	easy/medium/hard/v.hard
Hybrid mixed	full_mixed_scene	283	1599–3290	easy/medium/hard/v.hard
Hybrid mixed	polygons_plus_rings	250	1784–3222	easy/medium/hard/v.hard
Hybrid mixed	topology_plus_texture	270	1707–3510	easy/medium/hard/v.hard
Isotropic blobs	anisotropic_ellipses	150	1635–3386	easy/medium/hard/v.hard
Isotropic blobs	density_imbalanced_blobs	150	1575–3289	easy/medium/hard/v.hard
Isotropic blobs	imbalanced_cluster_sizes	150	1454–3509	easy/medium/hard/v.hard
Isotropic blobs	isotropic_blobs	150	1648–3278	easy/medium/hard/v.hard
Isotropic blobs	mixed_blob_ellipse	100	1751–3325	easy/medium/hard/v.hard
Isotropic blobs	outliers_and_noise_blobs	150	1854–3326	easy/medium/hard/v.hard
Isotropic blobs	overlapping_blobs	150	1847–3437	easy/medium/hard/v.hard
Ladder & grid	distorted_grid	200	1382–3363	easy/medium/hard/v.hard
Ladder & grid	grid_with_noise_and_gaps	194	1614–3448	easy/medium/hard/v.hard
Ladder & grid	missing_rungs_ladder	200	1765–3330	easy/medium/hard/v.hard
Ladder & grid	regular_grid	190	1615–3293	easy/medium/hard/v.hard
Ladder & grid	rotated_ladder	180	1652–3362	easy/medium/hard/v.hard
Ladder & grid	simple_ladder	190	1732–3306	easy/medium/hard/v.hard
Ladder & grid	sparse_lattice	180	1519–3223	easy/medium/hard/v.hard
Polygons	noisy_polygons	170	1822–3251	easy/medium/hard/v.hard
Polygons	open_frames	170	1559–3369	easy/medium/hard/v.hard

Family	Subfamily	Datasets	Points	Difficulty levels
Polygons	overlapping_polygons	170	1644–3302	easy/medium/hard/v.hard
Polygons	polygon_blob_mixture	177	1711–3347	easy/medium/hard/v.hard
Polygons	rectangles	160	1668–3379	easy/medium/hard/v.hard
Polygons	rotated_rectangles	160	1440–3304	easy/medium/hard/v.hard
Polygons	triangles_and_quadrilaterals	160	1505–3335	easy/medium/hard/v.hard
Polylines	branching_trees	220	1744–3361	easy/medium/hard/v.hard
Polylines	broken_piecewise_lines	200	1480–3259	easy/medium/hard/v.hard
Polylines	multi_segment_polylines	200	1611–3306	easy/medium/hard/v.hard
Polylines	noisy_corner_junctions	200	1839–3249	easy/medium/hard/v.hard
Polylines	sharp_corners	190	1710–3552	easy/medium/hard/v.hard
Polylines	two_segment_polylines	190	1810–3168	easy/medium/hard/v.hard
Polylines	y_junctions	200	1782–3359	easy/medium/hard/v.hard
Radial & spiral	broken_spirals	180	1466–3337	easy/medium/hard/v.hard
Radial & spiral	clean_spirals	180	1725–3432	easy/medium/hard/v.hard
Radial & spiral	meander_curves	180	1659–3376	easy/medium/hard/v.hard
Radial & spiral	noisy_meanders	187	1691–3302	easy/medium/hard/v.hard
Radial & spiral	radial_arc_mixture	180	1656–3232	easy/medium/hard/v.hard
Radial & spiral	radial_spokes	180	1561–3369	easy/medium/hard/v.hard
Radial & spiral	rotational_symmetry_patterns	180	1608–3357	easy/medium/hard/v.hard
Texture fields	anisotropic_grain	180	1273–3211	easy/medium/hard/v.hard
Texture fields	density_patchwork	187	1521–3330	easy/medium/hard/v.hard
Texture fields	fractal_dust	180	1410–3291	easy/medium/hard/v.hard
Texture fields	micro_blobs_texture	180	1550–3386	easy/medium/hard/v.hard
Texture fields	oriented_texture	180	1347–3240	easy/medium/hard/v.hard
Texture fields	speckle_fields	180	1705–3193	easy/medium/hard/v.hard
Texture fields	texture_plus_structure	180	1749–3435	easy/medium/hard/v.hard

B.3 Gallery and representative selection

Figure 5 shows one representative dataset from each family. The selection rule was declared before any plot was inspected and uses no performance quantity: for each family, take the `xy_2d` record whose frozen 250-dimensional label-free meta-feature vector, z-scored within the family, is closest in Euclidean distance to the family centroid, breaking ties on dataset identifier. The selected identifiers, the distances and the rule are recorded in `manifests/gallery_selection.json`. Panels are drawn on a common square window centered on the point cloud with equal aspect, so panel geometry does not vary with the data range; colors encode generator ground truth and are illustrative only.

C Complete method specification

Everything in this appendix describes frozen artifacts. Hyperparameters are those actually used to produce the reported results, not recommended settings.

C.1 Utility target

Equation (1) is implemented once and reused everywhere; its weight vector is fixed and was not tuned. In the notation of the main text, $G_j = (0.18\,\rho_j^+ + 0.12\,\tau_j^+ + 0.20\,\mathrm{PA}_j + 0.20\,\mathrm{NDCG}_j)/0.70$ and $T_j = (0.05\,\mathrm{OV}_j^{\mathrm{top}} + 0.05\,\mathrm{OV}_j^{\mathrm{bot}} + 0.10\,\mathrm{NDCG}_j^{\mathrm{top}} + 0.10\,\mathrm{NDCG}_j^{\mathrm{bot}})/0.30$. The eight terms are

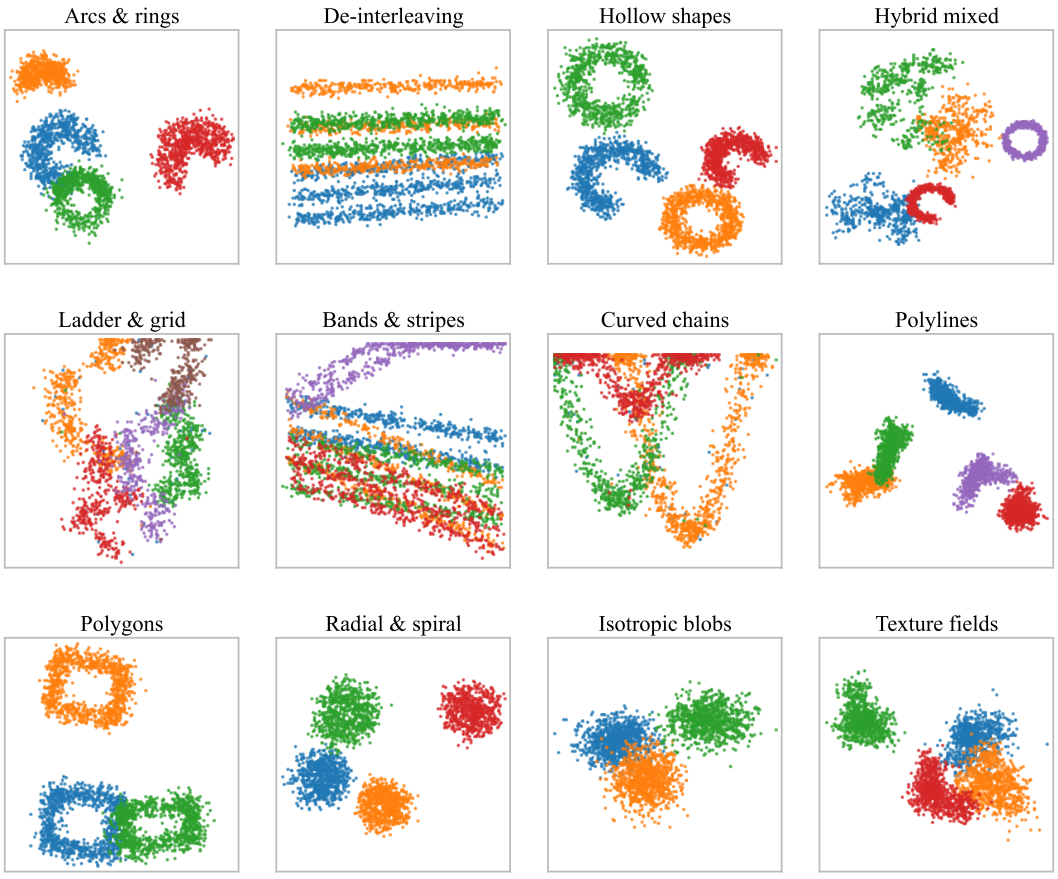

Figure 5: One representative dataset from each of the 12 structural families, colored by generator ground truth, selected by the meta-feature-centroid rule above. The families differ in the geometric property they vary: arcs and rings and hollow shapes are curvature-dominated, ladder and grid families are lattice-structured, bands and polylines are extended and piecewise-linear, de-interleaving superimposes independent sequences, and texture and density fields vary local density rather than shape. Because the representative is the family's meta-feature centroid rather than its most photogenic member, some panels look plainer than the family name suggests; that is a property of the rule, not a selection.

computed against the candidate ARI vector on the same (D, v), in this order: positive-part Spearman (0.18), positive-part Kendall (0.12), pairwise ordering accuracy over non-tied pairs (0.20), NDCG over the full candidate list with ARI as relevance (0.20), top-decile overlap (0.05), bottom-decile overlap (0.05), top-decile NDCG (0.10) and bottom-decile NDCG (0.10), the two bottom terms using $1 - \mathrm{ARI}$ as relevance. The top and bottom cut-offs are 10% of the candidate frame, so they are *frame-relative*: the candidate row basis is part of the measurement, which is why the candidate universe is derived from the candidate bank alone and never from index validity (Appendix T). An index is ineligible on (D, v) when fewer than $\max(10,\ 0.2Q)$ finite pairs are available; non-finite index values are ranked worst rather than dropped, and the result is clipped to $[0, 1]$. Targets are computed per (D, v), giving one 60-dimensional vector per record.

C.2 META-FEATURES

Each view is described by 250 label-free descriptors, organized in blocks: view-level shape and finiteness statistics; distributional summaries; geometric and structural descriptors; image-derived descriptors computed from a raster of the point cloud, including gray-level co-occurrence texture statistics; cross-space descriptors relating the view to the full representation; and landmarking features from cheap reference clusterings. No descriptor uses labels or the true cluster count. Feature extraction is per view, so one dataset yields three distinct vectors.

C.3 NEURAL UTILITY PREDICTOR

Multi-head network, $\mathbb{R}^{250} \to [0, 1]^{60}$. Shared trunk 512–512–256–256 with GELU activations, batch normalization and dropout $(0.15, 0.15, 0.10, 0.10)$; one head per index with hidden width 128, head dropout 0.05, sigmoid output. Loss $0.6\,\mathcal{L}_{\text{MSE}} + 0.25\,\mathcal{L}_{\text{rank}} + 0.15\,\mathcal{L}_{\text{top-10}}$, with pairwise margin 0.05, at most 256 sampled index pairs per sample, tie tolerance 10^{-4}, and top-k set to $k = 10$ with extra weight 2.0 inside the top-k term. Optimizer AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 512, up to 300 epochs, gradient-norm clipping at 1.0, plateau scheduler (factor 0.5, patience 8, minimum learning rate 10^{-6}), early stopping with patience 25, seed 42. Grouping is by `dataset_id`, inner validation ratio 0.15. Targets are the 60 `utility__` columns of the unified training table in a fixed order. Deployment uses the fold-ensemble checkpoint policy. The HEAD14 predictor is the identical architecture and schedule with output dimension 14.

C.4 NEAREST-NEIGHBOR UTILITY PREDICTOR

$\hat{\mathbf{u}}(\mathbf{z}) = \sum_{i \in \mathcal{N}_K} \omega_i \mathbf{u}_i / \sum_{i \in \mathcal{N}_K} \omega_i$ with $\omega_i = (d(\mathbf{z}, \mathbf{z}_i) + \epsilon)^{-1}$. The frozen configuration is $K = 15$ neighbors, standard scaling, Euclidean distance and inverse-distance weighting (configuration identifier `n015__standard__euclidean__inverse_distance`), with one index per view; each view index stores 16,013 training records over 250 features and 60 utility targets. The configuration was chosen by leave-one-split-out cross-validation on splits 2–16 under a one-standard-error rule: the threshold was 0.8447 against a best mean composite score of 0.8537, two configurations fell within one standard error, and the tie-break is smallest K, then lower query time, then simplicity. The data-dependent subset rule used by the `dynamic` policies is the frozen relative-softness heuristic `dynamic_topk_relsoft_a092_k5_ceil`.

C.5 POLICY SET AND POLICY SELECTORS

The frozen policy set is the full cross of {neural, nearest-neighbor} $\times$ {top-1, top-3, top-5, top-10, dynamic} $\times$ {raw normalized, softmax at temperature 0.5}, giving 20 policies (Table 13). Each policy is served by the reranker for its predictor $\times$ subset rule, so 20 policies map onto 10 rerankers. Because the weighting choice never changes which policy is selected — raw weighting wins the tie-break on every development record (Appendix J) — the portfolio is effectively the 10 predictor$\times$subset-rule variants, which is how the main text counts it.

Both policy selectors are extremely-randomized forests (`EXTRATREES__UNIFIED__META_UTILITIES__CENTERED`, configuration hash `a9e92d5fcfab1271`): 400 trees, minimum 8 samples per leaf, `max_features="sqrt"`, no scaling, `random_state` 42. Inputs are 373 features — the 250 meta-features of the view plus the predicted utility context — and the output is a 20-dimensional

score vector, one entry per policy; inference returns the $\arg\max$. Both are fitted on 48,039 development records from splits 2–16; Split 1 is used by neither. The leaf-size deviation from the nominal setting is recorded in the artifact manifest: 400 trees at minimum leaf size 2 measured 3.4 GB per forest on this training population, so leaf size 8 was used.

The two versions differ only in the target. PP V1 regresses the *centered original policy ARI*: for each record, the Best-View ARI each policy would have achieved from the $J_{\mathcal{S}}$-maximizer, centered within the record. PP V2 regresses the *centered reranker-aware* quantity: the Best-View ARI each policy would have achieved after reranking, built from single-exclusion out-of-fold reranker predictions so that the reranker scoring split s is never fitted on split s. Across the 48,039 records the $\arg\max$ of these two targets differs on roughly 69%, which is the measured reason retargeting matters.

C.6 CANDIDATE RERANKER

Ten scorers, one per regime (predictor $\times$ subset rule). Each is a `HistGradientBoostingRegressor` with squared-error loss, learning rate 0.08, 150 iterations, at most 31 leaves, minimum 20 samples per leaf, ℓ_2 regularization 1.0 and 255 bins, with a fixed seed; missing feature values are handled natively by the histogram splitter rather than imputed. Each is fitted on 347 features and $1{,}161{,}522$ candidate rows drawn from splits 2–16.

The feature vector $\phi(c, \mathcal{C}_{D,v}, D, v)$ has three groups. *Candidate features* describe the candidate alone: its configuration encoding, partition descriptors, and its raw and normalized index values. *Slate-context features* describe the slate it sits in: per-index location and spread across the slate, the candidate's rank under each index, and between-index agreement on the slate. *Dataset features* are the view's meta-features. Column admission is deny-by-default and semantic rather than substring-based: a column is a feature only if it is positively classified into one of the permitted roles, and any external-quality identifier, target column or oracle-derived column is forbidden. The target is the candidate's realized ARI; at deployment only the $\arg\max$ over the existing slate is evaluated. The scorer is thus a pointwise regressor used as a ranker; what makes its predictions slate-relative is the slate-context feature group (Appendix V.1).

Training rows are grouped by slate, and every out-of-fold target used to supervise PP V2 for split s comes from a reranker fitted on $\{2, \ldots, 16\} \setminus \{s\}$. Split 1 appears in none of the ten models. The inference procedure is: build the slate features for the view, score all eligible candidates with the reranker for the selected regime, return the highest-scoring candidate, and fall back to the $J_{\mathcal{S}}$-maximizer if no candidate is eligible.

D CONFIGURATION SEARCH SPACE

Three distinct candidate-generation scopes appear in this work and are never mixed.

1. Stage-1 fixed-budget diagnostic bank. The 32 candidate partitions already stored for a dataset view in the repository, re-scored under each arm's objective. Nothing is searched and nothing is sampled here: the arm reads the stored candidate table, applies its objective and returns a choice, so the recorded search runtime is zero and no seed applies. Because the candidate set is constant, only the objective varies, which is what makes the factorial of Section 5.1 an isolation of the criterion. This bank supports the fixed-budget arms of Figure 6 and Table 10, only.

Composition of the 32. Both banks resolve to the same deterministic enumeration of the space, identical for all three views: 4 k-means, 4 mini-batch k-means, 7 Gaussian mixture, 5 agglomerative, 5 Birch, 5 DBSCAN and 2 HDBSCAN configurations. The raw enumeration instead yields 6 Gaussian-mixture, 4 agglomerative, 4 Birch and 4 DBSCAN configurations together with 4 *constrained* HDBSCAN configurations; the constrained variant accepts a target cluster count, which cannot be supplied because the view object carries no ground truth, so those four slots are reassigned by fair round-robin to the four algorithms listed above, holding the bank at exactly 32. A candidate that fails keeps its slot and is never replaced, so the bank cannot silently grow its budget.

2. Online search. A Tree-structured Parzen sampler proposes configurations from the space in Table 7 and is scored by $J_{\mathcal{S}}$, with a budget of 50 evaluations and no early stopping. Candidates that fail or degenerate are retained in the log as failures rather than resampled, so a policy cannot be rewarded for producing unusable configurations. The measured runtime boundary opens when

the prepared array is handed to the arm and closes when the arm returns its chosen partition; corpus preparation lies outside it.

Seeding is not uniform across this work, and we state it per stage rather than assert one convention. No seed is ever derived from an outcome.

(a) **Stage-1 online factorial (controlled).** Seeded as

$$\mathrm{SHA256}\big(\texttt{42}\,|\,\texttt{dataset_id}\,|\,\texttt{view_id}\big),$$

reduced to 31 bits. The arm identity is *deliberately excluded* from the payload, so all six arms receive the same seed on the same dataset view and start from identical sampler randomness. This is what makes the factorial contrasts paired by construction.

(b) **Split-1 end-to-end online traces (controlled).** These predate the seeding work and were run on the historical unseeded path, that is, with the sampler's own non-deterministic initialization. They are the source of the pre- and post-reranker points, of Appendix H and of both policy selectors. The reranker contrast is unaffected, because the unreranked and reranked values come from the same run over the same visited slate; comparisons *between* separately searched policies on this split are not paired with respect to search randomness, which is the reason a superseded internal comparison was withdrawn (Appendix J).

(c) **Stage-4 external arms.** Seeded as

$$\mathrm{SHA256}\big(\texttt{1234}\,|\,\texttt{dataset_id}\,|\,\texttt{view_id}\,|\,\texttt{trace_identity}\big),$$

reduced to 31 bits, so arms that share a trace identity visit identical slates by construction and arms that do not are independently seeded.

Neither replay layer introduces randomness: the candidate reranker re-scores the candidates a trace actually visited, and the policy-selector evaluation is a lookup into the frozen per-policy outcome table, performing no clustering, no search and no reranker inference.

3. Stage-4 independent metric-analysis bank. A separate frozen bank of 32 candidate partitions per external dataset–view unit, constructed independently of the end-to-end executions and used only for the index analysis of Section 6. It is a *deterministic enumeration* of the same configuration space rather than a sample from it, so no seed applies to its construction; the four constrained-density configurations are replaced by fair round-robin to hold the bank at exactly 32. It shares no candidate with any method arm. Of its 150×32 slots, 4,776 carry a computable ARI; the remaining 24 failed and are excluded from the canonical candidate basis.

Space. Table 7 gives the space per view. Every k-based algorithm searches $k \in \{2, 3, 4, 5\}$; the true cluster count is never supplied. The `xy_2d` view additionally admits spectral clustering and a wider linkage and density range, because those methods are ill-posed or degenerate on a single coordinate. The four `hdbscan_constrained-`n blocks fix the component count to $n \in \{2, 3, 4, 5\}$ and search minimum cluster size, minimum samples and a persistence threshold; they are the constrained replacement used wherever a component-count-constrained density method is required. Generic index parameters are shared: noise points (label -1) are removed before scoring and a partition must have at least two clusters to be scored.

Table 7: The frozen configuration search space, per view. The true cluster count never enters: every k-based algorithm searches $k \in \{2, 3, 4, 5\}$. Constrained-HDBSCAN blocks fix the component count to n and search the remaining parameters.

View	Algorithm	Hyperparameters
`x_only`	`kmeans`	n_clusters $\in [2, 5]$; init $\in$ {k-means++}; n_init $= 10$; random_state $= 42$
	`minibatch_kmeans`	n_clusters $\in [2, 5]$; init $\in$ {k-means++}; batch_size $\in$ {100, 256}; random_state $= 42$
	`gmm`	n_components $\in [2, 5]$; covariance_type $\in$ {full, diag}; random_state $= 42$
	`agglomerative`	n_clusters $\in [2, 5]$; linkage $\in$ {ward, average, complete}
	`birch`	n_clusters $\in [2, 5]$; threshold $\in$ {0.05, 0.1, 0.2, 0.35}; branching_factor $\in$ {25, 50}

View	Algorithm	Hyperparameters
	dbscan	eps ∈ {0.03, 0.05, 0.08, 0.12, 0.2}; min_samples ∈ {3, 5, 10}
	hdbscan	min_cluster_size ∈ {10, 25, 50, 100}; min_samples ∈ {3, 5, 10}
	optics	min_samples ∈ {3, 5, 10}; xi ∈ {0.03, 0.05, 0.1}; min_cluster_size ∈ {0.02, 0.05, 0.1}
	hdbscan_constrained-2	n_components = 2; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-3	n_components = 3; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-4	n_components = 4; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-5	n_components = 5; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
y_only	kmeans	n_clusters ∈ [2, 5]; init ∈ {k-means++}; n_init = 10; random_state = 42
	minibatch_kmeans	n_clusters ∈ [2, 5]; init ∈ {k-means++}; batch_size ∈ {100, 256}; random_state = 42
	gmm	n_components ∈ [2, 5]; covariance_type ∈ {full, diag}; random_state = 42
	agglomerative	n_clusters ∈ [2, 5]; linkage ∈ {ward, average, complete}
	birch	n_clusters ∈ [2, 5]; threshold ∈ {0.1, 0.2, 0.35, 0.5}; branching_factor ∈ {25, 50}
	dbscan	eps ∈ {0.05, 0.1, 0.2, 0.35, 0.5}; min_samples ∈ {3, 5, 10}
	hdbscan	min_cluster_size ∈ {10, 25, 60, 120}; min_samples ∈ {3, 5, 10}
	optics	min_samples ∈ {3, 5, 10}; xi ∈ {0.03, 0.05, 0.1}; min_cluster_size ∈ {0.02, 0.05, 0.1}
	hdbscan_constrained-2	n_components = 2; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-3	n_components = 3; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-4	n_components = 4; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-5	n_components = 5; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
xy_2d	kmeans	n_clusters ∈ [2, 5]; init ∈ {k-means++}; n_init = 10; random_state = 42
	minibatch_kmeans	n_clusters ∈ [2, 5]; init ∈ {k-means++}; batch_size ∈ {100, 256}; random_state = 42
	gmm	n_components ∈ [2, 5]; covariance_type ∈ {full, diag, tied}; random_state = 42
	agglomerative	n_clusters ∈ [2, 5]; linkage ∈ {ward, average, complete, single}
	birch	n_clusters ∈ [2, 5]; threshold ∈ {0.1, 0.2, 0.35, 0.5, 0.8}; branching_factor ∈ {25, 50, 100}
	dbscan	eps ∈ {0.05, 0.1, 0.2, 0.3, 0.5, 0.8}; min_samples ∈ {3, 5, 10, 20}
	hdbscan	min_cluster_size ∈ {10, 25, 50, 100}; min_samples ∈ {3, 5, 10, 20}
	optics	min_samples ∈ {5, 10, 20}; xi ∈ {0.03, 0.05, 0.1}; min_cluster_size ∈ {0.02, 0.05, 0.1}
	spectral	n_clusters ∈ [2, 5]; affinity ∈ {nearest_neighbors, rbf}; n_neighbors ∈ {10, 20, 30}; random_state = 42
	hdbscan_constrained-2	n_components = 2; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-3	n_components = 3; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-4	n_components = 4; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}
	hdbscan_constrained-5	n_components = 5; min_cluster_size ∈ {25, 50}; min_samples ∈ {3, 5, 10}; persistence_thresh ∈ {0.0, 0.02, 0.05}

E UTILITY-PREDICTION EVALUATION

Table 8 reports every recorded quantity for the neural (multi-head) predictor with both inventories on the Split-1 records, which are excluded from learned-component fitting, and marks which are comparable across inventories; Table 9 breaks the neural FULL60 predictor down by view. The deployed CLUSTOPT (arm C4) uses the k-NN predictor instead. On the same 3,165 records it reaches MAE 0.095, RMSE 0.124, R^2 0.572, mean Spearman 0.713, Kendall 0.552, pairwise ranking accuracy 0.776, NDCG over all 60 indices 0.965 and NDCG@10 0.892, with top-1 accuracy 0.205 and top-5/top-10 overlap $0.437/0.564$: slightly below the neural predictor on every quantity.

The comparability caveat is worth stating once, precisely, because the raw numbers invite the wrong reading. Every top-k and NDCG@k quantity is computed over inventories of different size: $k = 10$ covers $10/14 = 71\%$ of HEAD14 but only $10/60 = 17\%$ of FULL60. HEAD14's top-10 overlap of 0.885 against FULL60's 0.605 therefore does not mean the smaller predictor is better at its job; it means the smaller task is easier by construction. The inventory-independent quantities are MAE, RMSE, R^2, Spearman, Kendall, pairwise ranking accuracy and NDCG over all indices. On those, the two neural predictors are close, with HEAD14 marginally better on error (0.0788 against 0.0831 MAE) and FULL60 marginally better on rank correlation (0.762 against 0.742). What matters for the pipeline is that the profile orders indices well enough to pick a useful top-k: for the deployed k-NN predictor, pairwise ranking accuracy 0.776 and NDCG over all 60 indices 0.965 on unseen data (0.800 and 0.971 for the neural one), with a gap to the oracle that Section 5.1 quantifies at the level of realized clustering quality.

Per view, the neural predictor is slightly weaker on the joint two-dimensional view (MAE 0.0905, NDCG@10 0.909) than on the single-coordinate views (MAE 0.0813 and 0.0774), which is the expected direction: `xy_2d` admits more distinct partition geometries and therefore a more variable utility profile.

Table 8: Utility-prediction quality on the Split-1 records, which are excluded from learned-component fitting ($n = 3{,}165$ dataset–view records). **The two columns are not comparable as a quality claim**: every top-k and NDCG@k quantity is computed over inventories of different size, so $k = 10$ covers 71% of HEAD14 but only 17% of FULL60. Inventory-independent quantities (MAE, RMSE, Spearman, pairwise accuracy, NDCG over all indices) are comparable.

Quantity	FULL60	HEAD14	Comparable?
MAE	0.0831	0.0788	yes
RMSE	0.1112	0.1069	yes
R^2	0.6569	0.7132	yes
Spearman ρ	0.7620	0.7425	yes
Kendall τ	0.6009	0.6226	yes
Pairwise ranking accuracy	0.8004	0.8113	yes
NDCG (all)	0.9711	0.9768	yes
NDCG@10	0.9101	0.9624	no
NDCG@5	0.9035	0.9429	no
NDCG@3	0.8987	0.9337	no
Top-10 overlap	0.6053	0.8847	no
Top-5 overlap	0.4785	0.7328	no
Top-3 overlap	0.3868	0.5917	no
Top-1 accuracy	0.2288	0.3852	no
Mean rank displacement	8.5149	1.9457	no

Table 9: FULL60 predictor quality by view.

View	MAE	Top-1 accuracy	NDCG@10	Records
`xy_2d`	0.0905	0.2227	0.9088	1055
`x_only`	0.0813	0.2227	0.8958	1055
`y_only`	0.0774	0.2408	0.9258	1055

F STAGE-1 FACTORIAL: FULL STATISTICS

Both regimes of the inventory×weighting factorial, with all six arm means and all pre-specified paired contrasts, their bootstrap intervals and their Holm-corrected decisions. Arm codes combine inventory (H for HEAD14, F for FULL60) with weighting (U uniform, P predicted, O oracle). The oracle arms are not deployable: they select the index subset in hindsight and exist to bound what better utility prediction could deliver.

The fixed-budget regime evaluates the 32-configuration diagnostic bank of Appendix D, so the candidate set is identical across all six arms. The online regime runs the 50-evaluation search under each arm's own objective, so the objective can also affect which configurations are visited. The criterion comparison of Appendix I separates endpoint from slate-oracle effects.

Table 10: Stage-1 factorial, fixed budget of 32 configurations. Arm codes: H/F = Head14 / Full60; U/P/O = uniform / predicted / oracle weighting.

Inventory	Weighting	Mean	95% interval	Deployable
head14	uniform	0.5014	[0.483, 0.520]	yes
head14	predicted	0.5890	[0.572, 0.606]	yes
head14	oracle	0.5997	[0.583, 0.616]	no (oracle)
full60	uniform	0.5021	[0.486, 0.519]	yes
full60	predicted	0.5966	[0.580, 0.613]	yes
full60	oracle	0.6250	[0.609, 0.641]	no (oracle)
Contrast	**Mean Δ**	**95% interval**	**Holm p**	**Rejected**
HP-HU	+0.0876	[+0.0789, +0.0966]	$< 10^{-4}$	yes
HO-HU	+0.0983	[+0.0894, +0.1072]	$< 10^{-4}$	yes
FU-HU	+0.0008	[−0.0101, +0.0117]	0.129	no
FP-HP	+0.0076	[+0.0022, +0.0135]	0.0773	no
FO-HO	+0.0253	[+0.0200, +0.0308]	$< 10^{-4}$	yes

Table 11: Stage-1 factorial, online search, 50 evaluations. Arm codes: H/F = Head14 / Full60; U/P/O = uniform / predicted / oracle weighting.

Inventory	Weighting	Mean	95% interval	Deployable
head14	uniform	0.3969	[0.378, 0.416]	yes
head14	predicted	0.5513	[0.533, 0.568]	yes
head14	oracle	0.5506	[0.533, 0.569]	no (oracle)
full60	uniform	0.3388	[0.321, 0.356]	yes
full60	predicted	0.5779	[0.561, 0.595]	yes
full60	oracle	0.5971	[0.581, 0.614]	no (oracle)
Contrast	**Mean Δ**	**95% interval**	**Holm p**	**Rejected**
HP-HU	+0.1544	[+0.1427, +0.1664]	$< 10^{-4}$	yes
HO-HU	+0.1536	[+0.1413, +0.1665]	$< 10^{-4}$	yes
FP-FU	+0.2392	[+0.2253, +0.2540]	$< 10^{-4}$	yes
FU-HU	−0.0582	[−0.0740, −0.0418]	$< 10^{-4}$	yes
FP-HP	+0.0266	[+0.0183, +0.0355]	$< 10^{-4}$	yes
FO-HO	+0.0466	[+0.0383, +0.0554]	$< 10^{-4}$	yes

G CLUSTOPT DESIGN EVOLUTION AND ARM CATALOG

This appendix exists so that the project's history is legible without polluting the main paper. It maps every scientifically relevant arm to its final shorthand, its mechanism and its status. Figure 7 gives the lineage in one picture: which stage fixed which component, and which arms the independent benchmark finally ran.

Stage 1 (vocabulary). Six arms on the fixed 32-configuration bank, the cross of inventory {HEAD14, FULL60} with weighting {uniform, predicted, oracle}, replicated under the online

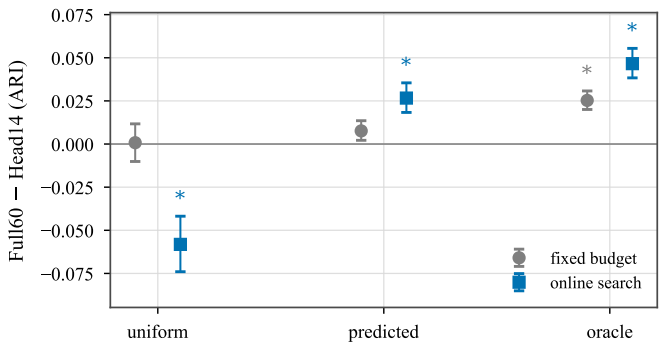

Figure 6: The FULL60−HEAD14 paired effect at each weighting strategy, under both regimes, with percentile bootstrap intervals. ∗ marks a contrast rejected under Holm correction within its pre-specified family. Enlarging the inventory is close to neutral at a fixed budget under uniform weighting and clearly harmful online, while the oracle over the same indices gains in both regimes.

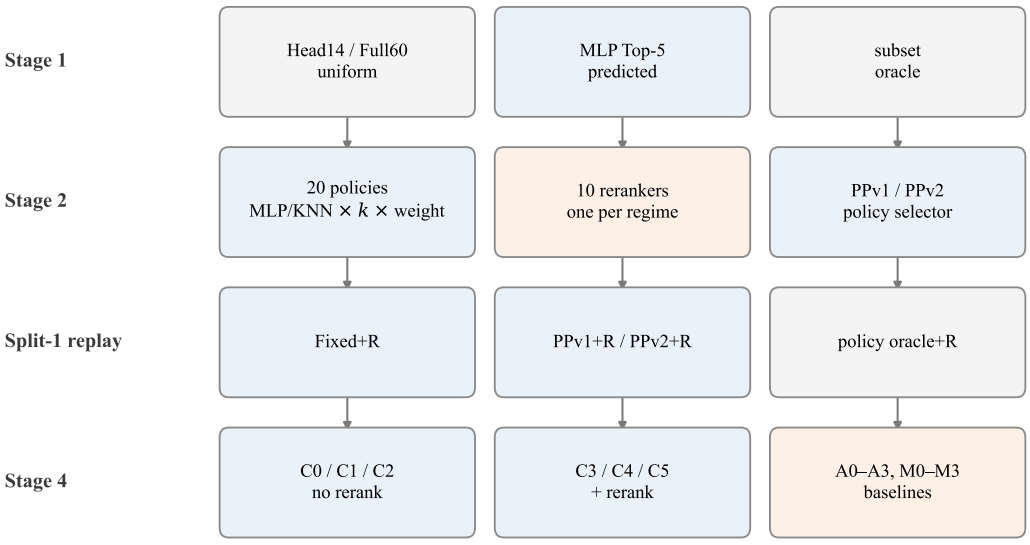

Figure 7: Arm lineage. Stage 1 establishes the inventory×weighting factorial on a fixed candidate bank. Stage 2 builds the deployable machinery: the 20-policy set, the ten regime-specific rerankers and the two policy selectors. The Split-1 replay evaluates the four-level progression. Stage 4 executes the six CLUSTOPT arms and the eight baseline arms on the independent benchmark.

regime. The predicted arms use the neural predictor with top-5 raw weighting; the oracle arms select the best index subset in hindsight and are *diagnostic*, not deployable. Authoritative source: `stage1_2x3_aggregation/{offline_fixed32,online_fixed50}`.

Stage 2 (policies, rerankers, selectors). The 20 policies of Table 13 are the cross of predictor $\times$ subset rule $\times$ weighting: `regressor_*` arms use the neural predictor, `knn_*` arms the nearest-neighbor predictor; subset rules are top-1, top-3, top-5, top-10 and dynamic; weighting is raw normalized or softmax at temperature 0.5. Ten rerankers serve them, one per predictor $\times$ subset rule. Two policy selectors are fitted, PP V1 against the pre-reranking endpoint and PP V2 against the reranker-aware endpoint. All are *deployable*. Authoritative sources: `candidate_reranker_split1_online_final` for the per-policy results, `stage2b5a_final_models_and_split1_preparation` for the reranker registry and the policy-to-reranker map.

Split-1 replay (the four levels plus the pre-reranker point). The deployable fixed policy is `knn_top10_raw`, chosen on splits 2–16 by Best-View mean and not on Split 1. Its unreranked and reranked results are the first two rows of Table 15; the reranked value reproduces the frozen four-level level A to machine precision, which is why the two are directly comparable. PP V1+R and PP V2+R are levels B and C. The reranker-assisted policy oracle is level D and is *diagnostic*. A separate hindsight-best fixed policy exists in the record and is also diagnostic; it happens to be the same policy, so no separate row is needed. Authoritative source: `stage2c_final_v2_split1_online_replay`.

Arms added after the external registry was frozen. The five criterion baselines (random search; uniform over FULL60; best single index on the development splits; static three-index core; dataset-independent profile $\bar{\mathbf{u}}$), AutoML4Clust and the corrected controlled ML2DAC run were specified after the 14 registered external arms had been unblinded. They are cataloged in Appendices I and Q and enter no pre-specified Holm family. All of them are *controlled*-split experiments: no post-hoc arm is reported on the primary external corpus, whose 14 registered arms are unchanged.

Stage 4 (external). Fourteen arms, frozen before any external execution and unchanged since. Table 12 lists them with their mechanisms. The mapping to the main text is: C0/C1/C2 return the J_S-maximizer and C3/C4/C5 return the reranker's choice, so C3 against C0 and C4 against C1 are exact reranker ablations. C4 is the system the main text calls CLUSTOPT; C5 adds the policy selector and is reported as a variant. A0–A3 and M0–M3 decompose each baseline's index inventory into original, +established, +new and extended. Authoritative source: `Independent_Domain_Benchmark/configs/method_registry.json` and the Stage-4C scientific review package.

Table 12: The 14 frozen external arms. Within each baseline family the only thing that varies is the index inventory, so that column is what distinguishes A0–A3 and M0–M3; live counts are the indices actually computable on this corpus, against 7/17/53/63 requested. Note the two inventories are not nested: a baseline's "Original" is the 7 classical indices, whereas HEAD14 and FULL60 contain only 4 of those 7 (Appendix L). Trace groups: C0/C3 share one physical search, C1/C4 share another, C2 and C5 each execute their own, A0–A3 share one, M0–M3 share one. Sharing is *within* a group only; no candidate slate is shared across systems.

Code	System	Index inventory (live)	Utility source	k	Policy	Rerank
C0	ClustOpt	FULL60 (60)	mlp	5	–	no
C1	ClustOpt	FULL60 (60)	knn	10	–	no
C2	ClustOpt	FULL60 (60)	policy	–	PP_v1	no
C3	ClustOpt	FULL60 (60)	mlp	5	–	MLP_TOP5
C4	ClustOpt	FULL60 (60)	knn	10	–	KNN_TOP10
C5	ClustOpt	FULL60 (60)	policy	–	PP_v2	per_policy
M0	ML2DAC	Original (7)	–	–	–	no
M1	ML2DAC	+Established (17)	–	–	–	no
M2	ML2DAC	+New46 (51)	–	–	–	no
M3	ML2DAC	Extended (61)	–	–	–	no
A0	AutoClust	Original (7)	–	–	–	no
A1	AutoClust	+Established (17)	–	–	–	no
A2	AutoClust	+New46 (51)	–	–	–	no
A3	AutoClust	Extended (61)	–	–	–	no

Table 13: The frozen policy set: 2 utility predictors × 5 subset rules × 2 weighting schemes = 20 policies, served by 10 rerankers (one per predictor × subset rule).

Policy	Predictor	Subset rule	Weighting
knn_top_dynamic_predictK_raw	KNN	dynamic	RAW
knn_top_dynamic_predictK_softmax_t05	KNN	dynamic	SOFTMAX_T05
knn_top1_raw	KNN	top1	RAW
knn_top1_softmax_t05	KNN	top1	SOFTMAX_T05
knn_top10_raw	KNN	top10	RAW
knn_top10_softmax_t05	KNN	top10	SOFTMAX_T05
knn_top3_raw	KNN	top3	RAW
knn_top3_softmax_t05	KNN	top3	SOFTMAX_T05
knn_top5_raw	KNN	top5	RAW
knn_top5_softmax_t05	KNN	top5	SOFTMAX_T05
regressor_top_dynamic_predictK_raw	MLP	dynamic	RAW
regressor_top_dynamic_predictK_softmax_t05	MLP	dynamic	SOFTMAX_T05
regressor_top1_raw	MLP	top1	RAW
regressor_top1_softmax_t05	MLP	top1	SOFTMAX_T05
regressor_top10_raw	MLP	top10	RAW
regressor_top10_softmax_t05	MLP	top10	SOFTMAX_T05
regressor_top3_raw	MLP	top3	RAW
regressor_top3_softmax_t05	MLP	top3	SOFTMAX_T05
regressor_top5_raw	MLP	top5	RAW
regressor_top5_softmax_t05	MLP	top5	SOFTMAX_T05

H CONTROLLED RESULTS: ALL POLICIES AND CONTRASTS

Table 14 reports all 20 frozen policies on the earlier unseeded controlled Split-1 replay before and after reranking, with the slate oracle for each. Every number in this appendix comes from that replay, not from the seeded criterion sweep of Table 1. Three things are visible in it that the main-text table cannot show.

First, reranking improves *every* policy, by between $+0.031$ and $+0.064$ ARI, with every interval excluding zero. The reranker's contribution is therefore not an artifact of the particular policy the main text uses.

Second, the gain is largest where the pre-reranker objective is weakest. The correlation between a policy's unreranked mean and its gain is -0.96 across the 20 policies, and the reranked means span 0.605–0.616 against 0.543–0.578 unreranked, so reranking compresses the spread by roughly a factor of three. This is the mechanism stated in Section 3: the reranker recovers what a mis-specified objective left on the slate, so it has more to recover when the objective is worse.

Third, the slate oracle varies little across policies (0.651 to 0.656), which says that the 20 policies visit slates of similar quality and differ mainly in which candidate they return. That is what makes final selection the right place to intervene, and it is why the four-level progression of Table 15 moves so little once reranking is present. Table 17 shows that the same holds for the non-conditioned criteria.

Table 14: All 20 frozen policies on the controlled Split-1 replay, before and after reranking. † marks the development-selected fixed policy used as the deployable baseline. Reranking improves every policy; the gain is largest where the pre-reranker objective is weakest.

Policy	No rerank	Reranked	Gain	95% interval	Slate oracle
knn_top10_raw†	0.5581	0.6160	+0.0579	[+0.050, +0.066]	0.6528
knn_top10_softmax_t05	0.5649	0.6159	+0.0511	[+0.043, +0.059]	0.6541
knn_top5_softmax_t05	0.5699	0.6150	+0.0451	[+0.038, +0.052]	0.6541
knn_top5_raw	0.5712	0.6143	+0.0431	[+0.037, +0.050]	0.6545
regressor_top10_raw	0.5757	0.6137	+0.0380	[+0.032, +0.044]	0.6511
knn_top_dynamic_predictK_softmax_t05	0.5673	0.6136	+0.0463	[+0.039, +0.053]	0.6541
regressor_top5_softmax_t05	0.5749	0.6134	+0.0385	[+0.032, +0.045]	0.6524
regressor_top5_raw	0.5784	0.6128	+0.0343	[+0.028, +0.040]	0.6526
knn_top_dynamic_predictK_raw	0.5702	0.6121	+0.0419	[+0.035, +0.049]	0.6558
regressor_top3_raw	0.5739	0.6118	+0.0378	[+0.031, +0.045]	0.6531
regressor_top3_softmax_t05	0.5736	0.6115	+0.0379	[+0.031, +0.045]	0.6516
knn_top3_raw	0.5576	0.6112	+0.0536	[+0.046, +0.062]	0.6531
regressor_top10_softmax_t05	0.5746	0.6110	+0.0364	[+0.030, +0.043]	0.6512
knn_top3_softmax_t05	0.5681	0.6108	+0.0427	[+0.035, +0.050]	0.6547
regressor_top_dynamic_predictK_raw	0.5770	0.6094	+0.0324	[+0.026, +0.039]	0.6515
regressor_top_dynamic_predictK_softmax_t05	0.5771	0.6084	+0.0313	[+0.025, +0.038]	0.6510
knn_top1_softmax_t05	0.5425	0.6070	+0.0645	[+0.056, +0.073]	0.6548
regressor_top1_raw	0.5558	0.6069	+0.0511	[+0.043, +0.060]	0.6528
regressor_top1_softmax_t05	0.5513	0.6067	+0.0554	[+0.048, +0.064]	0.6507
knn_top1_raw	0.5431	0.6047	+0.0616	[+0.053, +0.070]	0.6552

H.1 LEARNED POLICY SELECTION

The 20 policies differ in their per-dataset outcomes, so a per-view selector among them is a natural extension: a 400-tree extremely-randomized forest (373 inputs, 20 outputs, fitted out of fold on 48,039 development records; Appendix C). Its two versions differ only in the target: PP V1 targets the

Table 15: The four-level progression on the earlier *unseeded* controlled Split-1 replay (1,055 datasets, mean Best-View ARI, percentile bootstrap intervals); this is a different execution from the seeded criterion sweep of Table 1. The first two rows are the same policy on the same replay and isolate reranking exactly. The policy oracle is a ceiling, not a system.

Configuration	Mean	95% interval
Fixed policy (`knn_top10_raw`), no rerank	0.5581	[0.541, 0.575]
Fixed policy + rerank (= CLUSTOPT)	0.6160	[0.600, 0.633]
PP V1 + rerank	0.6078	[0.592, 0.624]
PP V2 + rerank	0.6171	[0.601, 0.633]
Policy oracle + rerank	0.6568	[0.641, 0.673]

best policy for the *search output*, PP V2 for the *reranked* output. The two targets disagree on roughly 69% of records, so a selector optimized for the pre-reranking endpoint optimizes the wrong quantity once a reranker is present. Training PP V2 against reranker-aware supervision is worth $+0.0093$ ARI over PP V1, interval $[+0.0052, +0.0136]$, $p = 1.8 \times 10^{-5}$, on 423 wins to 337 losses. The direction matters: PP V1 sits 0.008 *below* simply fixing one policy, recovering -20% of the available policy headroom, whereas PP V2 recovers $+2.6\%$. Against the fixed policy PP V2 gains $+0.0010$ with an interval spanning zero and 602 exact ties: it repairs PP V1's deficit but does not establish superiority. The reranked *policy* oracle — the hindsight-best policy per record, a portfolio ceiling rather than the candidate-level *slate* oracle — reaches 0.657, so the 0.040 ARI from PP V2 to it is *policy-selection* headroom across the evaluated portfolio; it exceeds PP V2 in all 12 structural families (Appendix K). Against AutoClust Extended under the matched search domain, PP V2+R differs by $+0.0026$, interval $[-0.0024, 0.0078]$, $p = 0.75$, with 433 wins, 195 ties and 427 losses. Externally (arm C5 against C4) the selector changes the mean by $+0.008$ with 40 of 50 datasets tied, at about 30% more runtime. We therefore deploy the fixed policy and report the selector as a negative result with measurable headroom.

I CRITERION BASELINES AND THE CONDITIONING ABLATION

Arms. All five comparators share CLUSTOPT's search space, TPE sampler, 50-evaluation budget, three views and per-dataset–view seeds, and differ only in J. (i) *Best single index on development*: the index with the highest mean true utility over splits 2–16, chosen without reference to Split 1 and resolved per view — `silhouette` on x, `noise_aware_silhouette` on y and `calinski_harabasz` on xy. (ii) *Static core*: uniform weights over the three generalist indices of Section 6. (iii) *Uniform*: uniform weights over all of FULL60. (iv) *Dataset-independent profile*: Equation (2) with $\hat{\mathbf{u}}$ replaced by $\bar{\mathbf{u}}$, the mean true utility profile over splits 2–16 computed per view type, with the same top-10 subset rule and the same raw weighting. This arm removes only the dependence of the criterion on (D, v). (v) *Random search*: 50 configurations drawn uniformly from the same space, with no criterion; its returned candidate is the one the sampler's own bookkeeping reports, and its slate oracle is what the 50 draws contained.

Reading. The strongest non-conditioned criterion is the dataset-independent profile $\bar{\mathbf{u}}$ at 0.5442, and CLUSTOPT exceeds it by $+0.0214$ $[+0.0128, +0.0300]$ with $p = 1.1 \times 10^{-10}$ and rank-biserial $+0.267$: that margin is the estimate of what conditioning on (D, v) contributes, holding the vocabulary, the subset size and the weighting rule fixed. The ordering of the arms is the same on the single xy view, where the gaps are larger ($\hat{\mathbf{u}}$ 0.4925 against $\bar{\mathbf{u}}$ 0.4220, $+0.0705$ $[+0.056, +0.085]$), because Best-View partly masks a criterion's mistakes by maximizing over three views. Two results qualify the headline. First, the margin over $\bar{\mathbf{u}}$ is an order of magnitude smaller than the $+0.2269$ that predicted weighting wins over uniform weighting of the same 60 indices: most of what the profile contributes is knowing which indices are worth using at all, and conditioning adds to that rather than accounting for it. Second, the deployed configuration is not the best conditioned configuration on this split: the Stage-1 neural profile with a top-5 subset reaches 0.5779, $+0.0123$ $[+0.0053, +0.0194]$ above C4. We deploy C4 because it is frozen, simpler and the arm carried unchanged into the external benchmark, not because it is the controlled optimum.

Per family. The conditioning gain is broad rather than concentrated. Against $\bar{\mathbf{u}}$, CLUSTOPT leads in 8 of the 12 structural families; six family-level bootstrap intervals exclude zero and none excludes

Table 16: Criterion baselines on the controlled Split 1, seeded sweep (1,055 datasets; online search, 50 evaluations per view; one seeded protocol). Each arm returns its own criterion's maximizer; no reranking is applied. Δ is the paired difference CLUSTOPT minus the arm on Best-View ARI, with a percentile bootstrap interval, wins/ties/losses (ties $|\Delta| < 0.001$) and a two-sided Wilcoxon p. The last two rows are *conditioned* reference arms reused from the Stage-1 factorial under the identical protocol, not comparators.

Criterion	Best-view	Single	Oracle	Δ	95% int.	W/T/L	p
Random search	0.4719	0.3983	0.6570	$+0.0937$	$[+0.082, +0.105]$	660/134/261	5.8e-49
Uniform, FULL60	0.3388	0.3059	0.6510	$+0.2269$	$[+0.212, +0.242]$	878/61/116	6.8e-127
Best single index (dev., per view)	0.5004	0.2796	0.6524	$+0.0652$	$[+0.055, +0.075]$	603/212/240	4.3e-41
Static three-index core	0.5283	0.4724	0.6538	$+0.0374$	$[+0.028, +0.047]$	508/250/297	2.0e-17
$\bar{\mathbf{u}}$, top-10	0.5442	0.4220	0.6531	$+0.0214$	$[+0.013, +0.030]$	463/313/279	1.1e-10
$\hat{\mathbf{u}}$, top-10 (CLUSTOPT)	0.5656	0.4925	0.6547	–	–	–	–
Stage-1 MLP top-5 (conditioned)	0.5779	0.5181	0.6549	-0.0123	$[-0.019, -0.005]$	328/371/356	0.054
Stage-1 oracle top-5 (ceiling)	0.5971	0.5388	0.6549	-0.0315	$[-0.040, -0.024]$	269/309/477	2.6e-18

zero in the opposite direction; the median family difference is $+0.0284$. It is largest on arcs and rings ($+0.0463$), and excluding that family the mean difference is still $+0.0181$, so it is not an arcs artifact. These are per-family percentile intervals with no multiplicity correction across the 12 families, and no family-level inference is claimed. The six whose intervals exclude zero are hollow shapes ($+0.0535$), bands and stripes ($+0.0521$), arcs and rings, polygons ($+0.0352$), curved chains ($+0.0347$) and lattices ($+0.0346$); the four negative point estimates — blobs (-0.0146), texture fields (-0.0132), polylines (-0.0125) and hybrid scenes (-0.0034) — all have intervals spanning zero. Against the static core the pattern is stronger still: CLUSTOPT leads in 11 of 12 families, six with intervals excluding zero, none in the opposite direction. Full per-family statistics, including the steering/selection split, are in the released result tables.

Where the difference arises: selection, not steering. Every arm's search reaches almost the same ceiling. Mean slate oracles span 0.6510 to 0.6570, a range of 0.0060, while returned endpoints span 0.3388 to 0.5971, a range of 0.2584 — a ratio of $43\times$. Writing each endpoint difference as $\Delta_{\text{endpoint}} = \Delta_{\text{oracle}} + \Delta_{\text{selection}}$, the selection term carries 92.8% of the difference against $\bar{\mathbf{u}}$, 97.7% against the static core, 96.6% against the best single index, 98.4% against uniform weighting and 102.5% against random search.

Table 17: Slate ceilings against returned endpoints, seeded sweep. "Steering" is the paired difference in slate oracle (what the search made available) and "selection" the remainder (what the criterion returned from it).

Criterion	Slate oracle	Endpoint	Gap to oracle	Steering	Selection
Random search	0.6570	0.4719	0.1851	-0.0024	$+0.0961$
Uniform, FULL60	0.6510	0.3388	0.3123	$+0.0036$	$+0.2232$
Best single index (dev.)	0.6524	0.5004	0.1521	$+0.0022$	$+0.0630$
Static three-index core	0.6538	0.5283	0.1256	$+0.0008$	$+0.0365$
$\bar{\mathbf{u}}$, top-10	0.6531	0.5442	0.1089	$+0.0015$	$+0.0199$
$\hat{\mathbf{u}}$, top-10 (CLUSTOPT)	0.6547	0.5656	0.0891	–	–

Random search is the diagnostic case: it has the *highest* mean slate oracle of any arm (0.6570, interval excluding zero against CLUSTOPT's) and the second-worst endpoint (0.4719). Fifty configurations drawn uniformly from the space already contain candidates as good as anything a conditioned search visits within the same budget; what differs between criteria is whether the system returns them. The endpoint-to-oracle gap therefore ranks the arms exactly as the endpoints do, from 0.0891 for CLUSTOPT to 0.3123 for uniform weighting. This replaces the search-steering explanation the draft gave, which these data do not support.

AutoML4Clust. Run on the controlled split with silhouette as its supplied index, its own four-algorithm k-center space and a 100-loop SMAC budget, it reaches 0.516 mean Best-View ARI on

Split 1. Because neither its search space nor its budget matches the arms above, it is reported here rather than in Table 1.

Adjusted mutual information. Repeating the headline contrasts with AMI in place of ARI: on the primary external corpus CLUSTOPT−AutoClust is $+0.043$ (interval $[-0.030, +0.127]$, $p = 0.80$) and CLUSTOPT−ML2DAC is $+0.062$ ($[+0.004, +0.147]$, $p = 0.026$), and the reranker ablation C4−C1 is $+0.072$ ($[+0.023, +0.132]$, $p = 7 \times 10^{-5}$) — the same conclusions as ARI in every case. On the controlled Split 1 the unreranked fixed policy trails matched AutoClust Extended by -0.040 ($[-0.046, -0.033]$) on AMI, against -0.056 on ARI. The corresponding *reranked* controlled contrasts are not computable from the stored artifacts: the per-trial traces store ARI but no per-trial labels or AMI, so AMI for a reranker-selected candidate requires re-running that configuration.

J NULL AND SUPERSEDED RESULTS

Raw against softmax weighting. An earlier, less controlled analysis suggested that learning whether to use raw normalized or softmax weights in Equation (2) was worth roughly $+0.012$ ARI. Under the frozen protocol it disappears entirely, and the argument does not depend on seeding: across all 48,039 development records the current tie-break selects raw weighting on *every* record, so the "learned weighting" arm and the always-raw arm are the same assignment and cannot differ. The earlier figure is attributable to the two arms having visited different candidates under separate unseeded searches (Appendix D) rather than to weighting. We retain the fixed raw scheme and report no weighting selector.

Data-dependent subset size. The policy set includes a data-dependent subset rule alongside the fixed top-1, top-3, top-5 and top-10 rules. It is retained as a policy because the selector may choose it, and it is chosen on about 10% of views (Table 14 shows its standalone performance), but it does not dominate the fixed rules and we make no claim for it.

The historical controlled baseline inventory arms. Both baselines were also run in all four index-inventory arms on the *controlled* repository, and those eight runs are frozen and exist. We show them in Table 18 rather than omit them, because a reader who finds them should be able to see that we knew about them and why they are not used. Two defects disqualify them as evidence of an inventory effect. First, the AutoClust arms are not paired: of 324 comparison keys, none had identical ordered candidate slates and all 324 differed, so the arms searched different candidates and any difference between them confounds the inventory with the search. Second, the ML2DAC arms carry that defect and a scorer defect — the predicted index was unevaluable for 54 of its 61 classifier classes, so every candidate tied and the incumbent stayed the top warm start, leaving the predicted index without influence on selection at all. An inventory effect measured under that scorer cannot be an inventory effect.

Accordingly **no inferential or causal claim in this paper relies on those eight arms**; they are discussed descriptively and nothing more. The historical controlled ML2DAC values are reported for completeness as legacy references in Table 18 only, and are used in no inferential or causal claim. The corrected scorer, produced for the external benchmark, has since been replayed on the controlled repository under the matched search domain, in both of the inventories the external benchmark reports: 0.5739 mean Best-View ARI with the +established inventory and 0.5663 with the extended one. The +established arm, which is the controlled counterpart of external arm M1, is the one Section 5.3 compares against. ML2DAC remains a first-class comparator on the external benchmark and in the runtime analysis, where all four of its inventory arms are properly paired and reported (Table 27).

It is worth recording that the two benchmarks disagree in direction. On the controlled arms the extended inventory looks best for AutoClust (0.6145 against 0.5810); on the properly paired external benchmark neither baseline's best arm is its extended one. That disagreement is itself the reason the unpaired ordering must not be read causally. None of this affects the cross-system contrast of Section 5.3, which also uses the extended AutoClust arm (0.6145): that contrast compares two systems, each on its own search trajectory over a matched domain, and never relied on slates being shared across AutoClust's inventory arms.

Table 18: Historical controlled baseline inventory arms (Stage 3A; 1,055 synthetic datasets, matched search domain, mean Best-View ARI). Shown for completeness because the runs exist and are frozen, and read as *legacy descriptive references only*. **They are not clean evidence of an index-inventory effect, and no inferential or causal claim in this paper relies on them.** † the AutoClust arms are *not paired*: 0 of 324 comparison keys had identical ordered candidate slates and 324/324 differed, so a difference between arms cannot be attributed to the inventory rather than to the different search each arm ran. ‡ the ML2DAC arms carry that defect *and* a scorer defect: the predicted index was unevaluable for 54 of 61 classifier classes, so it had no influence on selection at all. The properly paired replacement is the external benchmark (Table 2), where the ordering is different: there neither baseline's best arm is its extended one.

Arm	Index inventory	Requested	Live	Best-View mean ARI
A0 †	Original	7	7	0.5810
A1 †	+Established	17	17	0.6047
A2 †	+New46	53	51	0.6063
A3 †	Extended	63	61	0.6145
M0 ‡	Original	7	7	0.5721
M1 ‡	+Established	17	17	0.5739
M2 ‡	+New46	53	51	0.5688
M3 ‡	Extended	63	61	0.5663

K CONTROLLED PER-FAMILY ANALYSIS

Table 19 gives the absolute per-family means behind Figure 8, and Figure 9 shows the same values as a heatmap so that family difficulty, which dominates the absolute scale, is visible separately from the between-system differences.

Family difficulty spans a wide range: polylines and polygons sit above 0.81 mean ARI for every configuration, while parallel bands, lattices, curved chains and de-interleaving sit below 0.44 for every configuration. That spread is much larger than any between-system difference, which is precisely why the main text plots differences rather than levels.

Two descriptive observations follow. The comparator throughout this appendix, including the `AutoClust` column of Table 19, is *AutoClust Extended* under the matched search domain; no other AutoClust arm appears here. For the deployed configuration (the "Fixed+R" column) the point estimate is positive in 4 families and negative in 8; the largest difference in either direction is 0.031, on isotropic blobs in AutoClust's favor; and 6 of the 12 differences lie within ± 0.01 ARI. De-interleaving, at $+0.013$, lies just outside that band. Only three family-level bootstrap intervals exclude zero — arcs and rings ($+0.030$) and de-interleaving ($+0.013$) in CLUSTOPT's favor, isotropic blobs (-0.031) in AutoClust's — and the other nine are unresolved rather than equal. We therefore do not describe the positive signs as four wins: the sign count and the size of the differences say different things, and the sizes are what matter. The ± 0.01 band is a descriptive reading aid, not an equivalence threshold, and no equivalence test was performed. Separately, and at the level of the policy portfolio rather than of any single slate, the reranked policy oracle exceeds PP V2 in every one of the 12 families by between 0.022 and 0.064 (Appendix H), so the remaining *policy-selection* headroom is distributed across geometries rather than concentrated in one.

No family-level inference is claimed anywhere. These are descriptive means over the frozen per-dataset Split-1 rows; the families were not a pre-specified testing family and no correction was applied to them.

L INTERNAL METRIC ANALYSIS: GENERALISTS AND SPECIALISTS

L.1 DEFINITIONS, FROZEN BEFORE COMPUTATION

Generalist core. An index belongs to the core if it is in the top five *both* by true utility *and* by selection frequency in at least six of the 12 families. The threshold, the k and the double requirement were declared before the quantities were computed. Exactly three indices qualify — `silhouette` (Original), `noise_aware_silhouette` (Established) and `gridness_fft_acf` (New46) — and

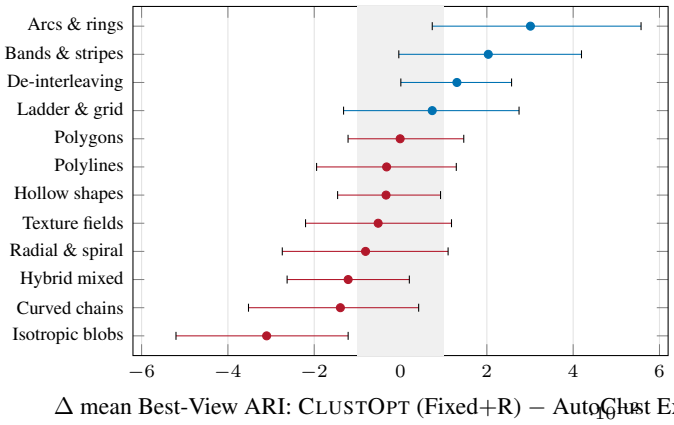

Figure 8: Per-family difference from matched *AutoClust Extended* on the controlled replay, for the deployed configuration (Fixed+R), ordered by that difference with 95% bootstrap intervals. The shaded band marks differences of at most 0.01 ARI. Descriptive; no family-level inference is claimed.

Table 19: Controlled per-family means on the earlier unseeded Split-1 replay (descriptive; no family-level inference is claimed). The comparator is *AutoClust Extended* under the matched search domain, and families are ordered by the deployed CLUSTOPT (Fixed+R) difference against it. Intervals are percentile bootstraps over datasets; W/T/L counts ties at $|\Delta| < 0.001$.

Family	n	Fixed+R	AutoClust	Δ vs AC	95% interval	W/T/L
Arcs & rings	116	0.7695	0.7394	+0.0301	[+0.0074, +0.0557]	41/43/32
Bands & stripes	98	0.3402	0.3199	+0.0203	[−0.0004, +0.0419]	47/9/42
De-interleaving	102	0.3978	0.3847	+0.0131	[+0.0001, +0.0257]	54/20/28
Ladder & grid	83	0.3775	0.3702	+0.0073	[−0.0132, +0.0275]	43/6/34
Polygons	67	0.8156	0.8157	−0.0001	[−0.0122, +0.0146]	20/24/23
Polylines	91	0.8199	0.8231	−0.0032	[−0.0195, +0.0129]	27/32/32
Hollow shapes	91	0.7861	0.7895	−0.0034	[−0.0146, +0.0093]	29/27/35
Texture fields	79	0.7001	0.7053	−0.0052	[−0.0220, +0.0118]	29/16/34
Radial & spiral	73	0.7118	0.7199	−0.0081	[−0.0274, +0.0110]	29/14/30
Hybrid mixed	125	0.6463	0.6584	−0.0121	[−0.0263, +0.0020]	43/20/62
Curved chains	79	0.3713	0.3852	−0.0139	[−0.0352, +0.0042]	28/11/40
Isotropic blobs	51	0.7299	0.7609	−0.0310	[−0.0520, −0.0121]	15/11/25

Against matched *AutoClust Extended* the point estimate is positive in 4 families and negative in 8; 6 of the 12 differences lie within ±0.01 ARI; the largest in either direction is 0.0310 on isotropic blobs. Only three intervals exclude zero: arcs and rings and de-interleaving in CLUSTOPT's favor, isotropic blobs against. Signs are reported descriptively, not as wins.

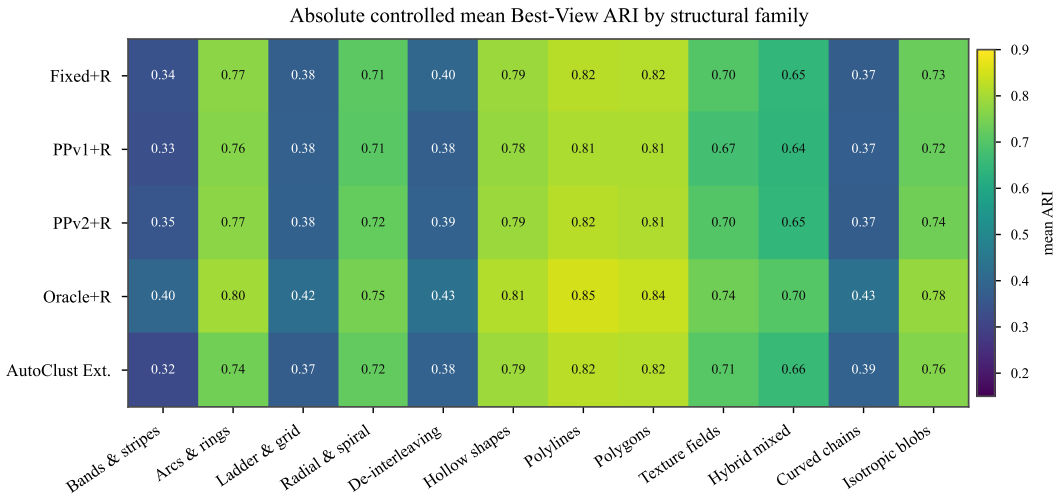

Figure 9: Absolute controlled mean Best-View ARI by structural family and configuration. Family difficulty dominates the color scale, which is why the main-text panel shows differences instead.

the set is unchanged at thresholds of 5, 6 and 7 of 12, with one index from each provenance at every threshold. Table 22 reports the per-family alignment behind the selection half of the criterion.

Specialization tiers. For an index m and a group g (a family or a subfamily), $\Delta U = U_g(m) - U_{\bar{g}}(m)$ is the utility uplift inside the group against a comparator. At family level the comparator is all datasets outside the family; at subfamily level it is the *sibling subfamilies of the same family*, which is the stricter test because it holds the broad geometry fixed and asks whether the index discriminates *within* it. A relation is **Tier A** when $\Delta U > 0$, its bootstrap interval excludes zero, and it survives Benjamini–Hochberg correction within its pre-specified test family. **Tier B** relations meet the first two conditions but not the third. Enrichment is the ratio of in-group to out-of-group selection rate and is reported descriptively.

L.2 What the definitions find

At family level, 14 Tier-A relations survive correction, spanning 10 of the 12 families. 10 of the 14 carry the `unique_specialist` flag of the frozen relation table — the index is Tier A in that family and in no other — and those 10 fall in 8 families; five further relations are Tier B. Every family has at least one new-index specialist relation at some tier. At subfamily level, all 86 subfamilies are eligible and 42 Tier-A relations survive, covering 32 subfamilies, with 36 further Tier-B relations.

The number that matters most is 27: that many Tier-A subfamily relations are *invisible at family level*. They concern an index that is strongly discriminative on one subfamily and unremarkable across the parent family as a whole. This is the sharpest internal evidence for the paper's premise, because it identifies the granularity at which the useful signal lives — and it is finer than the granularity at which any of the compared systems conditions.

Specialist sets are also largely disjoint between families: the mean pairwise Jaccard overlap of specialist sets across the 66 family pairs is 0.15 with a median of 0, against 0.50 for top-5 utility sets and 0.59 for top-5 selection sets. Families therefore agree substantially on which indices are broadly good and disagree almost completely on which are specifically good, which is exactly the structure a dataset-conditioned selector can exploit and a globally shared criterion does not explicitly represent.

L.3 Pattern-based against image-based new indices

The deployed FULL60 inventory is 4 classical indices, 10 established variants and the 46 newly designed structural indices; HEAD14 is the first two groups.

The *designed* inventory of the 46 is **22 pattern-based and 24 image-based**, as recorded in the frozen metric registry and set out index by index in Appendix W. The subgroup comparison below

has a smaller denominator, and the reason is worth stating exactly rather than describing the two indices as unlabeled. `convexity_ratio_image_based` and `hough_arc_circle_strength` are image-based by design and are live in CLUSTOPT (finite utility targets on all 48,039 development records), but in the frozen Stage-3 matrices they appear under their internal names (`convexity_solidity_image`, `hough_circle_arc_strength`) and carry no subgroup label. The internal pattern-versus-image comparison therefore runs over $22 + 22 = 44$ indices, not 46. They are not all-NaN: the empty columns in the baselines' index matrix, which left the baselines' realized counts at 51 and 61 rather than 53 and 63, come from a name-lookup defect in the baseline path (Appendix A). Both are also fully evaluable on the external corpus (coverage 1.00; available-case utility 0.4032 and 0.3892), so the gap is internal to this one analysis and not a property of the design.

Among the 44 compared internally, mean utilities of the two subgroups are close (0.463 image, 0.453 pattern), but the image-based subgroup carries more Tier-A family relations (eight against six) and more top-5 appearances (17 against 7). Neither subgroup is redundant.

Table 20: All 14 Tier-A family-level specialisation relations. ΔU is the utility uplift of the index inside the family against the same index outside it; q is the Benjamini–Hochberg-adjusted value within the pre-specified test family. Enrichment is the ratio of in-family to out-of-family selection rate.

Family	Index	n	ΔU	95% interval	q	Enrichment
Arcs & rings	`connected_components_count`	1584	$+0.0527$	$[+0.047, +0.059]$	$< 10^{-4}$	1.64
Arcs & rings	`contour_graph_connectivity`	1584	$+0.0514$	$[+0.045, +0.058]$	$< 10^{-4}$	1.87
Hollow shapes	`axial_symmetry`	1309	$+0.0557$	$[+0.050, +0.062]$	$< 10^{-4}$	1.50
Hybrid mixed	`axial_symmetry`	1908	$+0.0244$	$[+0.019, +0.030]$	$< 10^{-4}$	1.63
Hybrid mixed	`connected_components_count`	1908	$+0.0220$	$[+0.017, +0.028]$	$< 10^{-4}$	1.48
Ladder & grid	`gridness_fft_acf`	1251	$+0.0519$	$[+0.046, +0.058]$	$< 10^{-4}$	1.30
Bands & stripes	`ribbon_thickness`	1402	$+0.0674$	$[+0.061, +0.074]$	$< 10^{-4}$	2.62
Polylines	`connected_components_count`	1309	$+0.0589$	$[+0.052, +0.065]$	$< 10^{-4}$	2.01
Polygons	`axial_symmetry`	1100	$+0.1007$	$[+0.095, +0.106]$	$< 10^{-4}$	1.61
Radial & spiral	`gridness_fft_acf`	1194	$+0.0314$	$[+0.023, +0.039]$	$< 10^{-4}$	1.17
Isotropic blobs	`axial_symmetry`	949	$+0.0934$	$[+0.088, +0.099]$	$< 10^{-4}$	1.50
Isotropic blobs	`gridness_fft_acf`	949	$+0.0341$	$[+0.025, +0.043]$	$< 10^{-4}$	1.33
Texture fields	`axial_symmetry`	1188	$+0.0679$	$[+0.062, +0.074]$	$< 10^{-4}$	1.91
Texture fields	`gridness_fft_acf`	1188	$+0.0473$	$[+0.040, +0.055]$	$< 10^{-4}$	1.19

Table 21: All 42 Tier-A subfamily-level specialisation relations; the comparator is the sibling subfamilies of the same family. "Sub-only" marks the 27 relations that are invisible at family level.

Subfamily	Family	Index	n	ΔU	95% interval	Sub-only
`annuli_variable_thickness`	Arcs & rings	`axial_symmetry`	185	$+0.1015$	$[+0.086, +0.116]$	yes
`full_rings`	Arcs & rings	`axial_symmetry`	185	$+0.1072$	$[+0.095, +0.121]$	yes

Subfamily	Family	Index	n	ΔU	95% interval	Sub-only
nested_overlapping_arc_pairs	Arcs & rings	fractal_dimension	188	+0.1690	[+0.157, +0.181]	yes
nested_overlapping_arc_pairs	Arcs & rings	ribbon_thickness	188	+0.1672	[+0.150, +0.184]	yes
nested_overlapping_arc_pairs	Arcs & rings	parallel_bands	188	+0.0996	[+0.088, +0.111]	yes
nested_overlapping_arc_pairs	Arcs & rings	glcm_haralick	188	+0.0943	[+0.082, +0.107]	yes
noisy_overlapping_arcs	Arcs & rings	gridness_fft_acf	260	+0.0406	[+0.020, +0.060]	yes
partial_circles	Arcs & rings	connected_components_count	186	+0.0274	[+0.011, +0.046]	–
partial_circles	Arcs & rings	contour_graph_connectivity	186	+0.0342	[+0.017, +0.054]	–
amp_arc_trends	De-interleaving	gridness_fft_acf	237	+0.1382	[+0.128, +0.151]	yes
hybrid_deinterleaving_scene	De-interleaving	gridness_fft_acf	248	+0.0807	[+0.065, +0.095]	yes
pri_amp_coupled	De-interleaving	gridness_fft_acf	241	+0.1296	[+0.116, +0.143]	yes
pri_parallel_trains	De-interleaving	curvature_consistency	228	+0.0277	[+0.014, +0.039]	yes
pri_with_jitter	De-interleaving	curvature_consistency	227	+0.0241	[+0.012, +0.037]	yes
bridged_components	Hollow shapes	axial_symmetry	190	+0.0434	[+0.029, +0.058]	–
broken_hollow_shapes	Hollow shapes	connected_components_count	186	+0.0210	[+0.004, +0.038]	yes
broken_hollow_shapes	Hollow shapes	contour_graph_connectivity	186	+0.0379	[+0.019, +0.059]	yes
single_hole_rings	Hollow shapes	axial_symmetry	178	+0.0254	[+0.011, +0.041]	–
bands_plus_blobs	Hybrid mixed	axial_symmetry	234	+0.0244	[+0.013, +0.038]	–
fragmented_band_with_overlapping_clouds	Hybrid mixed	axial_symmetry	189	+0.0713	[+0.059, +0.084]	–
fragmented_band_with_overlapping_clouds	Hybrid mixed	connected_components_count	189	+0.0342	[+0.018, +0.049]	–
fragmented_band_with_overlapping_clouds	Hybrid mixed	convexity_ratio	189	+0.0464	[+0.033, +0.061]	yes
topology_plus_texture	Hybrid mixed	gridness_fft_acf	252	+0.0239	[+0.006, +0.041]	yes
distorted_grid	Ladder & grid	gridness_fft_acf	184	+0.0321	[+0.019, +0.045]	–
regular_grid	Ladder & grid	gridness_fft_acf	177	+0.0323	[+0.018, +0.046]	–
crossing_or_near_crossing_bands	Bands & stripes	gridness_fft_acf	205	+0.1242	[+0.109, +0.140]	yes
crossing_or_near_crossing_bands	Bands & stripes	ribbon_thickness	205	+0.1331	[+0.119, +0.148]	–
crossing_or_near_crossing_bands	Bands & stripes	convexity_ratio	205	+0.1097	[+0.095, +0.124]	yes
thick_ribbons	Bands & stripes	gridness_fft_acf	185	+0.1221	[+0.108, +0.135]	yes
thick_ribbons	Bands & stripes	ribbon_thickness	185	+0.1099	[+0.095, +0.126]	–
arc_parabola_mixture	Curved chains	gridness_fft_acf	170	+0.0375	[+0.025, +0.051]	yes

Subfamily	Family	Index	n	ΔU	95% interval	Sub-only
curves_with_density_drift	Curved chains	gridness_fft_acf	166	+0.0762	[+0.063, +0.088]	yes
clean_spirals	Radial & spiral	axial_symmetry	170	+0.0587	[+0.043, +0.073]	yes
noisy_meanders	Radial & spiral	gridness_fft_acf	178	+0.0437	[+0.026, +0.059]	–
radial_arc_mixture	Radial & spiral	connected_components_count	168	+0.0576	[+0.040, +0.072]	yes
radial_spokes	Radial & spiral	connected_components_count	170	+0.0177	[+0.001, +0.038]	yes
rotational_symmetry_patterns	Radial & spiral	axial_symmetry	171	+0.0575	[+0.042, +0.072]	yes
mixed_blob_ellipse	Isotropic blobs	connected_components_count	94	+0.0316	[+0.010, +0.053]	yes
overlapping_blobs	Isotropic blobs	gridness_fft_acf	141	+0.0638	[+0.046, +0.082]	–
micro_blobs_texture	Texture fields	gridness_fft_acf	168	+0.0238	[+0.006, +0.040]	–
oriented_texture	Texture fields	connected_components_count	168	+0.0625	[+0.042, +0.083]	yes
speckle_fields	Texture fields	gridness_fft_acf	170	+0.0500	[+0.030, +0.070]	–

Table 22: Alignment between an index's true utility and how often CLUSTOPT selects it, within each family (Spearman over the 60 indices of the deployed inventory), together with how much of the true top-5 the deployed selector recovers.

Family	n	ρ all	ρ New46	Top-5 recall	Top-1 hit
Arcs & rings	1584	0.819	0.798	0.607	0.761
De-interleaving	1631	0.726	0.758	0.481	0.628
Hollow shapes	1309	0.843	0.838	0.614	0.786
Hybrid mixed	1908	0.882	0.860	0.556	0.673
Ladder & grid	1251	0.866	0.778	0.601	0.759
Bands & stripes	1402	0.755	0.737	0.410	0.532
Curved chains	1188	0.862	0.811	0.604	0.754
Polylines	1309	0.843	0.807	0.611	0.772
Polygons	1100	0.841	0.860	0.596	0.736
Radial & spiral	1194	0.791	0.756	0.621	0.762
Isotropic blobs	949	0.853	0.854	0.638	0.788
Texture fields	1188	0.855	0.852	0.593	0.739
Mean		0.828	0.809	0.578	0.724

M WHAT THE SYSTEM ACTUALLY SELECTS, BY FAMILY

Figure 10 shows the family×index selection structure for a transparently chosen subset of indices. The subset rule is stated here and was not adjusted after inspection: take the three core indices, then every index carrying Tier-A or Tier-B family-level specialization evidence, in that order. No index was chosen by hand, and no index was dropped for looking uninformative.

The display is row-centered — each index's selection rate has its own across-family mean subtracted — *for visualization only*, because selection rates differ by an order of magnitude between the core and the specialists and an uncentered heatmap shows only that difference. The underlying uncentered rates and the corresponding family-conditional utilities are the `S_group` and `U_group` columns of the frozen Stage-3 outputs and will be included among the released derived result tables as `family_metric_selection_matrix.csv` and `family_metric_utility_matrix.csv`.

Three readings are visible in the figure and are consistent with Table 22. The core indices are selected broadly: under the deployed top-10 configuration on Split 1 they fill 21.5% of all selected slots,

Table 23: Pattern-based against image-based new indices, pooled over the 12 families.

Subgroup	Indices	Mean utility	Selection share	Tier-A relations
New46-Image	22	0.4632	0.2496	8
New46-Pattern	22	0.4534	0.2256	6

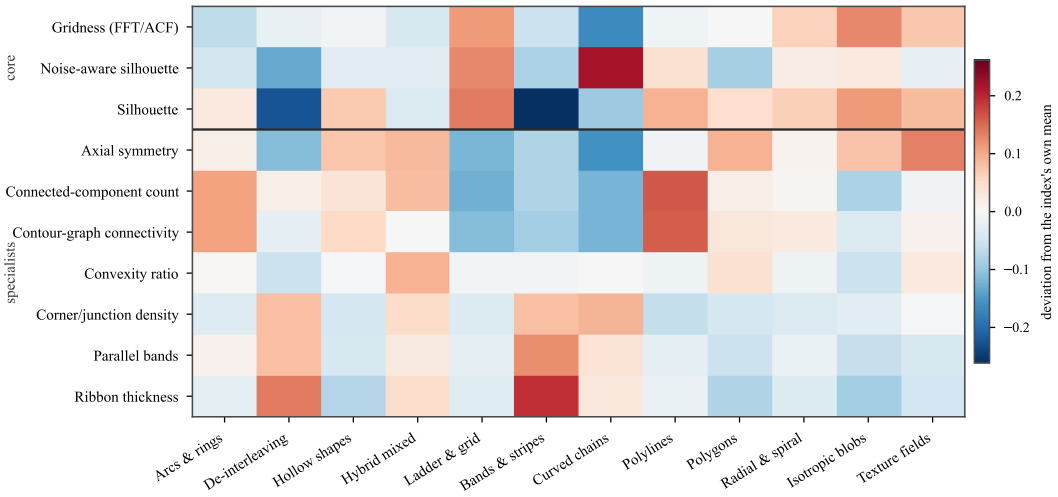

Figure 10: Which indices CLUSTOPT selects, by family: in-family selection rate, centered on each index's own across-family mean so that localization is visible. Rows above the rule are the three-index generalist core; rows below carry family-level specialization evidence. The centring is a display choice only; raw rates are in the frozen Stage-3 outputs.

against 50.9% for the 46 new indices and the remainder for the classical and established indices outside the core. With ten slots per record, these shares are not comparable with those of a five-index subset. Their row-centered deviations are small, which is what being a generalist means. The specialist rows behave in the opposite way: each is bright in a few family columns and flat elsewhere. And the columns where a specialist is bright are the geometries it was designed for — lattice and band descriptors on grids and stripes, symmetry and connected-component descriptors on rings and hollow shapes, ribbon and thickness descriptors on de-interleaving.

For the deployed configuration (k-NN profile, top-10, raw weighting) on the 3,165 Split-1 dataset–view records, the within-family Spearman correlation between an index's true utility and its selection frequency averages 0.929 over the 12 families, ranging from 0.895 on ladders and grids to 0.955 on hybrid mixed geometry, and 0.891 restricted to the new indices. Averaged over families, the ten selected indices recover 0.65 of the true top-5 and contain the true top-1 on 0.72 of records, falling to 0.55 top-5 recall on the new indices. Table 22 reports the same quantities for a different estimator — a top-5 subset rule on the neural profile over the development splits (mean $\rho = 0.828$, top-5 recall 0.58, top-1 hit 0.72). Because the two rules select five and ten indices respectively, the higher recall and correlation of the deployed rule are partly mechanical and are not evidence of better alignment. In both, the shortfall is largest where the vocabulary is newest, which is consistent with part of the remaining predicted-to-oracle gap of Section 5.1 arising from imperfect identification or exploitation of the newer specialist vocabulary. We report this as an observed pattern and do not claim to have localized the gap.

Which policy, by family. Table 24 asks the same question one level up: which *policy* PP V2 selects for each family, over the 3,165 controlled records. The overall preference is for the nearest-neighbor predictor with a top-10 subset (0.37 of records), followed by the neural predictor with top-10 (0.23), but the distribution is markedly family-dependent. On polylines it is concentrated, 0.51 on a single regime; on de-interleaving and on parallel bands and stripes it is diffuse, with no regime above 0.25 and noticeably more weight on small subsets and on the data-dependent rule. That is the same signal as the index-level panel, expressed in the units the selector actually chooses in: where a geometry admits one reliable objective the selector commits to it, and where a family is

internally heterogeneous it spreads. The two diffuse families are also the two where subfamily-level specialization is densest (Appendix L). This qualitative pattern is consistent with the finer-grained specialization observed there, although we do not treat the distribution of policy choices and the density of subfamily specialization as a tested association.

Table 24: What PP V2 selects, by structural family: the fraction of the family's dataset–view records assigned to each policy regime on the controlled Split-1 replay (3,165 records). Regimes name the predictor and the subset rule; the weighting is raw throughout (Appendix J). Rows sum to one.

Family	knnt10	mlpt10	mlpt3	knnt3	knnt5	mlpdyn	mlpt5	knnt1	knndyn	mlpt1
Arcs & rings	0.38	0.26	0.07	0.09	0.02	0.04	0.03	0.03	0.03	0.03
De-interleaving	0.19	0.21	0.09	0.06	0.07	0.13	0.08	0.04	0.07	0.06
Hollow shapes	0.45	0.21	0.06	0.05	0.04	0.04	0.03	0.05	0.04	0.03
Hybrid mixed	0.44	0.22	0.05	0.04	0.05	0.06	0.07	0.02	0.04	0.02
Ladder & grid	0.46	0.18	0.08	0.05	0.06	0.06	0.04	0.02	0.03	0.02
Bands & stripes	0.19	0.25	0.07	0.02	0.10	0.07	0.11	0.07	0.05	0.07
Curved chains	0.26	0.26	0.07	0.02	0.08	0.07	0.08	0.06	0.09	0.01
Polylines	0.51	0.18	0.05	0.06	0.03	0.02	0.04	0.04	0.07	0.01
Polygons	0.40	0.27	0.10	0.04	0.03	0.03	0.03	0.04	0.03	0.01
Radial & spiral	0.38	0.27	0.07	0.06	0.05	0.06	0.02	0.06	0.01	0.01
Isotropic blobs	0.35	0.19	0.08	0.08	0.07	0.05	0.05	0.05	0.03	0.05
Texture fields	0.43	0.23	0.04	0.09	0.06	0.04	0.04	0.03	0.02	0.03
All	0.37	0.23	0.07	0.06	0.05	0.06	0.05	0.04	0.04	0.03

N CASE STUDY: DE-INTERLEAVING

De-interleaving is worth one page because it is the clearest case in the repository where compactness and separation are the wrong question, and because it is not a family on which CLUSTOPT clearly wins: the controlled difference from matched AutoClust is $+0.013$, small, with a bootstrap interval ($[+0.0001, +0.026]$) that only just excludes zero, and no family-level inference is claimed (Appendix K). It therefore illustrates the mechanism without doubling as a result.

What the structure is. A de-interleaving dataset superimposes several independent periodic sequences in a two-dimensional measurement space — in the repository's parameterization, time of arrival against a second observed quantity. Each source contributes its own regularly spaced train of points; the observed cloud is the union. The correct partition assigns each point to its originating source. The seven subfamilies vary what makes that hard: `pri_parallel_trains` superimposes regular trains, `pri_with_jitter` perturbs the spacing, `missing_pulses` deletes points, `spurious_pulses_noise` adds points belonging to no source, `amp_arc_trends` and `pri_amp_coupled` make the second coordinate drift with the first, and `hybrid_deinterleaving_scene` combines them.

Why compactness-oriented indices misjudge it. The correct clusters are interleaved in both coordinates: two points that are nearest neighbors in the plane usually belong to *different* sources, because consecutive arrivals alternate between trains. On these constructions a criterion built on within-cluster tightness against between-cluster separation can therefore favor locally adjacent mixtures over source-consistent partitions, which makes such criteria poorly matched to the de-interleaving objective. The observed difficulty of the family is consistent with that failure mode: it is the hardest family in the repository for every system, with means between 0.38 and 0.40 against above 0.81 for polylines and polygons (Table 19).

What the analysis actually shows. Five Tier-A specialization relations survive correction on this family, all at subfamily level and *all invisible at family level*. On `amp_arc_trends` and `pri_amp_coupled` the specialist is `gridness_fft_acf`, whose utility uplift against sibling subfamilies is $+0.138$ and $+0.130$ with in-subfamily selection enriched by factors of 1.48 and 1.51; on `hybrid_deinterleaving_scene` the same index gains $+0.081$. On `pri_parallel_trains` and `pri_with_jitter` the specialist is instead `curvature_consistency`, with smaller uplifts ($+0.028$, $+0.024$) but higher selection enrichment (1.66, 1.38). The switch between the two specialists tracks the switch in structure: where the second coordinate drifts, a periodicity descriptor computed from the autocorrelation of the rasterized cloud is informative, and where the trains are straight and regular a curvature-consistency descriptor is.

That is the paper's premise at its finest observable granularity. A single criterion for the family cannot express it, because the two halves of the family want different indices; and a family-level analysis cannot even see it, because neither relation survives when the seven subfamilies are pooled. Consistently, PP V2's policy choice on this family is the most diffuse of the 12 (Table 24).

O THE INDEPENDENT EXTERNAL CORPUS

Fifty datasets from three sources, assembled and frozen before any system was run. Each was cleaned, imputed where required, scaled and projected once to a two-dimensional array whose checksum is recorded; all 50 recorded checksums match the frozen manifest, whose own hash is `709712107424ad4e`. Twenty-nine required a PCA projection because their cleaned dimensionality exceeded two; the remainder are two-dimensional as distributed. Ground truth is stored in a separate file that the scientific loader never returns.

Sources. Seven FCPS shape benchmarks, retrieved by HTTP with the raw bytes checksummed. Fifteen scikit-learn generator datasets: five generators (anisotropic blobs, concentric circles, moons, varied-density blobs and a standard blob mixture) at three difficulty levels each, with the generator parameters and seed recorded; this is the structure that gives the corpus 40 dependence groups rather than 50, because the three difficulty variants of one generator are not independent observations. Twenty-eight real tabular datasets, 26 of them projected by PCA and two (`cmc`, `haberman`) natively two-dimensional after cleaning: 27 retrieved from OpenML by numeric data identifier, and one (Breast Cancer Wisconsin (Diagnostic), a UCI dataset licensed under CC BY 4.0) loaded through scikit-learn 1.6.0.

Redistribution. We do not redistribute raw or prepared external datasets. Table 25 provides the source identifier, preparation metadata and checksum needed to reconstruct and verify each prepared representation. The released loaders apply the documented preparation pipeline and verify the resulting representation against the recorded checksum.

Table 25: N.1 — the independent external corpus: reproducibility metadata. "Identifier" is what reconstructs the dataset: an OpenML data id for real tabular sources, the generator with its difficulty level and seed for the scikit-learn shapes, and the named FCPS suite otherwise. K is the number of ground-truth classes. "Dim" is the dimensionality after cleaning and before the projection to two dimensions; PCA is applied wherever that exceeds two, and the recorded total explained variance is given with it. "Checksum" is the first eight hex characters of the frozen two-dimensional representation's checksum, which every system loaded. External raw and prepared data are not redistributed; the released loaders reconstruct each dataset and verify the prepared representation against this checksum. Per-dataset results are in Table 26.

Dataset	Source	Identifier	n	K	Dim	PCA (EV)	Checksum
`fcps_atom`	FCPS	FCPS suite	800	2	3	yes (0.68)	`00a531cc`
`fcps_chainlink`	FCPS	FCPS suite	1000	2	3	yes (0.67)	`94a9e7bf`
`fcps_engytime`	FCPS	FCPS suite	4096	2	2	no	`96f09b35`
`fcps_lsun`	FCPS	FCPS suite	400	3	2	no	`3ae873c7`
`fcps_tetra`	FCPS	FCPS suite	400	4	3	yes (0.68)	`792dc578`
`fcps_twodiamonds`	FCPS	FCPS suite	800	2	2	no	`2fcc2fd4`
`fcps_wingnut`	FCPS	FCPS suite	1016	2	2	no	`34ccfe98`
`real_analcatdata_authorship`	real (PCA)	OpenML 458	841	4	70	yes (0.26)	`cf3f8720`
`real_balance_scale`	real (PCA)	OpenML 11	625	3	4	yes (0.50)	`c3c0f14c`
`real_banknote`	real (PCA)	OpenML 1462	1372	2	4	yes (0.87)	`aa23d1be`
`real_blood_transfusion`	real (PCA)	OpenML 1464	748	2	4	yes (0.91)	`71d60699`
`real_breast_cancer_wdbc`	real (PCA)	scikit-learn 1.6.0 / UCI 17	569	2	30	yes (0.63)	`e4bc97b1`
`real_car_evaluation`	real (PCA)	OpenML 21	1728	4	21	yes (0.14)	`c96e6ef9`
`real_cardiotocography_3c`	real (PCA)	OpenML 1560	2126	3	35	yes (0.32)	`a7bc7136`
`real_churn`	real (PCA)	OpenML 40701	5000	2	16	yes (0.25)	`5d991722`
`real_climate_model_crashes`	real (PCA)	OpenML 1467	540	2	20	yes (0.16)	`c769a386`

Dataset	Source	Identifier	n	K	Dim	PCA (EV)	Checksum
real_cmc	real (PCA)	OpenML 23	1473	3	2	no	b0027095
real_credit_approval	real (PCA)	OpenML 29	690	2	6	yes (0.50)	1063bf33
real_diabetes_pima	real (PCA)	OpenML 37	768	2	8	yes (0.48)	618aa914
real_german_credit	real (PCA)	OpenML 31	1000	2	7	yes (0.44)	fce34db0
real_haberman	real (PCA)	OpenML 43	306	2	2	no	91dfb7cb
real_hill_valley	real (PCA)	OpenML 1479	1212	2	100	yes (0.99)	b8d7b1c9
real_ilpd_liver	real (PCA)	OpenML 1480	583	2	9	yes (0.53)	dfddb517
real_ionosphere	real (PCA)	OpenML 59	351	2	33	yes (0.40)	75681f21
real_kc1_software	real (PCA)	OpenML 1067	2109	2	21	yes (0.82)	ff6d939e
real_kc2_software	real (PCA)	OpenML 1063	522	2	21	yes (0.89)	2ab0b219
real_madelon	real (PCA)	OpenML 1485	2600	2	500	yes (0.02)	32c52259
real_ozone_level_8hr	real (PCA)	OpenML 1487	2534	2	72	yes (0.62)	df7dbb63
real_pc1_software	real (PCA)	OpenML 1068	1109	2	21	yes (0.75)	fb79323b
real_pc3_software	real (PCA)	OpenML 1050	1563	2	37	yes (0.60)	8c4c6084
real_pc4_software	real (PCA)	OpenML 1049	1458	2	37	yes (0.53)	2073d596
real_pollen	real (PCA)	OpenML 871	3848	2	5	yes (0.71)	d8bd4025
real_qsar_biodeg	real (PCA)	OpenML 1494	1055	2	41	yes (0.30)	ddd593a5
real_seeds	real (PCA)	OpenML 1499	210	3	7	yes (0.89)	0c4d4a13
real_sonar	real (PCA)	OpenML 40	208	2	60	yes (0.39)	707f911b
sk_aniso_blobs_easy	scikit-learn	aniso_blobs, easy, seed 1234	1500	3	2	no	96ccc183
sk_aniso_blobs_hard	scikit-learn	aniso_blobs, hard, seed 1236	1500	3	2	no	1c3dc7b9
sk_aniso_blobs_medium	scikit-learn	aniso_blobs, medium, seed 1235	1500	3	2	no	bf948367
sk_blobs_easy	scikit-learn	blobs, easy, seed 1334	1500	4	2	no	cac7e19a
sk_blobs_hard	scikit-learn	blobs, hard, seed 1336	1500	4	2	no	287559f1
sk_blobs_medium	scikit-learn	blobs, medium, seed 1335	1500	4	2	no	dcbec0f4
sk_circles_easy	scikit-learn	circles, easy, seed 1434	1500	2	2	no	ec92e465
sk_circles_hard	scikit-learn	circles, hard, seed 1436	1500	2	2	no	94df3f08
sk_circles_medium	scikit-learn	circles, medium, seed 1435	1500	2	2	no	a67c1aa2
sk_moons_easy	scikit-learn	moons, easy, seed 1534	1500	2	2	no	8cc0cd5c
sk_moons_hard	scikit-learn	moons, hard, seed 1536	1500	2	2	no	f9b50630
sk_moons_medium	scikit-learn	moons, medium, seed 1535	1500	2	2	no	efe3dd02
sk_varied_blobs_easy	scikit-learn	varied_blobs, easy, seed 1634	1500	3	2	no	9e382ca0
sk_varied_blobs_hard	scikit-learn	varied_blobs, hard, seed 1636	1500	3	2	no	6d16fdb4
sk_varied_blobs_medium	scikit-learn	varied_blobs, medium, seed 1635	1500	3	2	no	215387ee

Table 26: N.2 — headline per-dataset results. Best-View ARI on each external dataset for the deployed system and the three reference arms. Metadata for the same datasets is in Table 25.

Dataset	Source	C5	C4	A0	M1
fcps_atom	FCPS	0.532	0.893	0.732	0.520
fcps_chainlink	FCPS	0.099	0.099	0.187	0.156
fcps_engytime	FCPS	0.488	0.488	0.817	0.472
fcps_lsun	FCPS	1.000	1.000	0.824	0.251
fcps_tetra	FCPS	0.315	0.315	0.342	0.315

Dataset	Source	C5	C4	A0	M1
fcps_twodiamonds	FCPS	1.000	1.000	1.000	0.970
fcps_wingnut	FCPS	1.000	1.000	0.759	1.000
real_analcatdata_authorship	real (PCA)	0.740	0.740	0.663	0.827
real_balance_scale	real (PCA)	0.098	0.098	0.121	0.121
real_banknote	real (PCA)	0.215	0.215	0.240	0.144
real_blood_transfusion	real (PCA)	0.031	0.031	0.059	0.031
real_breast_cancer_wdbc	real (PCA)	0.659	0.659	0.690	0.497
real_car_evaluation	real (PCA)	0.092	0.092	0.092	0.029
real_cardiotocography_3c	real (PCA)	0.027	0.027	0.123	0.102
real_churn	real (PCA)	0.014	0.014	0.010	0.004
real_climate_model_crashes	real (PCA)	0.021	0.021	0.010	0.018
real_cmc	real (PCA)	0.017	0.017	0.024	0.019
real_credit_approval	real (PCA)	0.002	0.002	0.130	0.078
real_diabetes_pima	real (PCA)	0.116	0.116	0.153	0.079
real_german_credit	real (PCA)	0.072	0.072	0.073	0.027
real_haberman	real (PCA)	0.136	0.136	0.146	0.007
real_hill_valley	real (PCA)	0.004	0.000	0.001	0.000
real_ilpd_liver	real (PCA)	0.039	0.014	-0.038	-0.004
real_ionosphere	real (PCA)	0.146	0.146	0.140	0.143
real_kc1_software	real (PCA)	0.011	0.003	0.241	0.059
real_kc2_software	real (PCA)	0.044	0.044	0.328	0.011
real_madelon	real (PCA)	0.028	0.028	0.057	0.005
real_ozone_level_8hr	real (PCA)	-0.014	-0.005	0.006	-0.001
real_pc1_software	real (PCA)	0.166	0.166	0.152	0.166
real_pc3_software	real (PCA)	0.120	0.120	0.165	0.050
real_pc4_software	real (PCA)	0.129	0.129	0.117	0.017
real_pollen	real (PCA)	-0.000	-0.000	0.000	0.000
real_qsar_biodeg	real (PCA)	0.189	0.034	0.036	-0.001
real_seeds	real (PCA)	0.785	0.497	0.609	0.503
real_sonar	real (PCA)	0.022	0.022	0.040	0.079
sk_aniso_blobs_easy	scikit-learn	1.000	1.000	1.000	0.997
sk_aniso_blobs_hard	scikit-learn	0.571	0.571	0.571	0.567
sk_aniso_blobs_medium	scikit-learn	0.979	0.979	0.992	0.755
sk_blobs_easy	scikit-learn	1.000	1.000	0.922	0.966
sk_blobs_hard	scikit-learn	0.352	0.352	0.539	0.352
sk_blobs_medium	scikit-learn	0.483	0.643	0.852	0.925
sk_circles_easy	scikit-learn	1.000	1.000	0.147	-0.001
sk_circles_hard	scikit-learn	0.101	0.101	0.037	-0.000
sk_circles_medium	scikit-learn	0.899	0.899	0.077	0.000
sk_moons_easy	scikit-learn	1.000	1.000	0.357	0.985
sk_moons_hard	scikit-learn	0.839	0.839	0.479	0.816
sk_moons_medium	scikit-learn	0.981	0.981	0.393	0.993
sk_varied_blobs_easy	scikit-learn	0.796	0.796	0.687	0.722
sk_varied_blobs_hard	scikit-learn	0.946	0.571	0.834	0.719
sk_varied_blobs_medium	scikit-learn	0.661	0.567	0.747	0.704

P ALL EXTERNAL ARMS AND ALL PAIRED CONTRASTS

Table 27 reports all 14 external arms on both accuracy scopes together with the source-balanced macro and the median three-view runtime. Table 28 reports every pre-specified paired contrast on Best-View ARI with its group-aware bootstrap interval, Wilcoxon value, rank-biserial effect size and win/tie/loss counts.

Two features of the arm table are worth pointing at. The inventory decompositions are not monotone for either baseline: AutoClust's best variant uses its original indices and both extended variants are worse, and ML2DAC's best variant is the +established one rather than the extended one. And the source-balanced macro, which weights the three corpora equally instead of by dataset count, compresses the ranking substantially and reorders the two CLUSTOPT variants, which is why we present them as operating points rather than as a ranking.

P.1 CONCENTRATION OF THE ADVANTAGE OVER AUTOCLUST

The paired CLUSTOPT−AutoClust (C4−A0) difference on the primary corpus, $+0.0370$ over 50 datasets, is not spread across datasets. By source it is $+0.1775$ on the 15 scikit-learn generator datasets, $+0.0190$ on FCPS and -0.0338 on the 28 real tabular datasets, where the group-bootstrap interval $[-0.0643, -0.0083]$ excludes zero. Within the generators, the concentric-circle and two-moon groups (`skgen_circles`, `skgen_moons`: six datasets, two dependence groups) contribute

$+0.0666$ to the 50-dataset mean difference. Leaving those two groups out, CLUSTOPT averages 0.3343 and AutoClust 0.3680 on the remaining 44 datasets, a difference of -0.0337 with group-bootstrap interval $[-0.0654, -0.0012]$ (38 dependence groups), Wilcoxon $p = 0.026$, rank-biserial -0.40 and 12 wins, 4 ties and 28 losses for CLUSTOPT (12/6/26 with ties at $|\Delta| < 0.001$). The interval excludes zero in AutoClust's favor: the positive overall mean difference against AutoClust is concentrated in the concentric-circle and two-moon groups, and outside those groups AutoClust performs better on average. For the policy-selector variant C5 the corresponding leave-out difference is -0.0242 (interval $[-0.0609, +0.0112]$). By contrast, the difference from ML2DAC's best arm (M1) is positive in all three sources ($+0.159$ FCPS, $+0.120$ scikit-learn, $+0.015$ real) and $+0.0667$ over all 50 datasets, interval $[+0.0012, +0.1571]$, $p = 0.039$, with 30 wins, 5 ties and 15 losses. We report the concentration because a mean difference carried by two generator groups is a statement about non-convex, connectivity-defined structure, not about clustering in general. Whether it extends beyond those two generators is untested: it needs a corpus of natively two-dimensional non-convex benchmarks, which we have not assembled.

Table 27: All 14 external arms on the independent benchmark.

Code	Arm	Best-View	95% interval	Single	Src-bal.	Runtime
C5	Policy Predictor v2 + reranker	0.3990	[0.260, 0.521]	0.3284	0.5156	62.9
C4	KNN Top-10 RAW + reranker	0.3906	[0.254, 0.512]	0.3189	0.5203	48.3
A0	original CVIs	0.3536	[0.232, 0.467]	0.3206	0.4661	48.4
A2	original + New46	0.3495	[0.220, 0.470]	0.3092	0.4678	130.1
C3	MLP Top-5 RAW + reranker	0.3488	[0.234, 0.455]	0.2719	0.4656	41.4
A3	extended CVIs	0.3316	[0.211, 0.444]	0.3058	0.4414	133.1
M1	original + Established	0.3239	[0.184, 0.455]	0.2229	0.4224	20.2
C2	Policy Predictor v1, no reranker	0.3174	[0.195, 0.431]	0.2554	0.4310	66.9
C0	MLP Top-5 RAW, no reranker	0.3127	[0.209, 0.409]	0.2523	0.4142	41.3
A1	original + Established	0.3085	[0.204, 0.410]	0.2497	0.4063	51.8
C1	KNN Top-10 RAW, no reranker	0.3076	[0.190, 0.418]	0.1948	0.4235	47.2
M3	extended CVIs	0.3007	[0.183, 0.413]	0.1804	0.3932	20.8
M0	original CVIs	0.2859	[0.174, 0.391]	0.1493	0.3949	23.1
M2	original + New46	0.2731	[0.168, 0.372]	0.1454	0.3546	21.9

Table 28: Every *distinct* pre-specified external paired contrast on Best-View ARI; intervals resample the 40 dependence groups. The frozen comparison file lists `C5 PPv2_R vs C2 PPv1_noR` twice, once under a second interpretive label; it is one numerical comparison and appears here once. Note that it changes the policy target *and* adds reranking, so it is not an isolated reranker effect: the isolated reranker contrasts are C3−C0 and C4−C1.

Contrast	Mean Δ	95% interval	p	W/T/L	Rank-biserial
C5 PPv2_R vs C4 KNN_R	$+0.0083$	$[-0.018, +0.036]$	0.432	7/40/3	0.309
C5 PPv2_R vs C3 MLP_R	$+0.0501$	$[-0.002, +0.101]$	0.351	18/15/17	0.184
C5 PPv2_R vs C2 PPv1_noR	$+0.0816$	$[+0.012, +0.166]$	0.0215	28/10/12	0.415
reranker: C3 vs C0 (MLP Top-5)	$+0.0361$	$[-0.019, +0.092]$	0.202	24/10/16	0.234
reranker: C4 vs C1 (KNN Top-10)	$+0.0830$	$[+0.021, +0.159]$	0.00127	27/13/10	0.590
C5 vs best AutoClust (A0)	$+0.0453$	$[-0.039, +0.141]$	0.931	21/4/25	0.016
C5 vs best ML2DAC (M1)	$+0.0751$	$[+0.009, +0.166]$	0.00511	32/5/13	0.472
C4 vs best AutoClust (A0)	$+0.0370$	$[-0.047, +0.134]$	0.626	18/4/28	-0.084
C4 vs best ML2DAC (M1)	$+0.0667$	$[+0.001, +0.157]$	0.0391	30/5/15	0.353

Q BASELINE IMPLEMENTATIONS AND ADDITIONAL SYSTEMS

What each baseline learned from. AutoClust's algorithm-selection k-NN and its CVI→ARI surrogate, and ML2DAC's meta-knowledge repository and CVI classifier, were fitted on splits 2–16 of the controlled repository under the same three view definitions (`x_only`, `y_only`, `xy_2d`), using 48,039 records (16,013 datasets $\times$ 3 views). Neither saw Split 1 or any external dataset, and neither was retrained for the external corpus. AutoClust's algorithm selector is a 1-nearest-neighbor classifier over its seven CVI meta-features (missing values imputed as 0), and its surrogate an MLP regressor (hidden layers 60–30–10, ReLU, at most 300 iterations, early stopping on a 10% validation fraction) over the arm's index inventory. ML2DAC's meta-knowledge is a random-forest classifier (100 trees, Gini) over 52 meta-features together with a KD-tree (leaf size 40) warm-start index. Both use seed 1234. Deviations from the published descriptions, beyond those forced by the shared search space and index inventories, have not been cataloged.

Re-implementation and fidelity. Both baselines were re-implemented so that they could consume the shared prepared arrays, views, configuration space and index inventories; their selection logic follows the published descriptions (Poulakis et al., 2020; Treder-Tschechlov et al., 2023). We have *not* validated either re-implementation against the authors' own code: ML2DAC has a public implementation and the natural check — running it with its own search space and meta-knowledge on our external corpus and comparing it with our re-implementation — has not been performed, and for AutoClust we have neither run official code nor reproduced a headline number from the original paper. Any discrepancy between our baseline results and the originals could therefore be a property of our re-implementation rather than of the method, and we flag that as a limitation of the comparison rather than resolving it.

AutoClust on concentric circles. On `sk_circles_easy` AutoClust returns a partition of ARI 0.147 although the configuration space contains partitions of ARI 1.0, and this one dataset is part of why the source-level difference on the scikit-learn generators is large. The main text attributes this to a compactness-weighted global surrogate selecting the wrong partition. That attribution is an interpretation of the method's design, not a measurement: the supporting diagnostic — which configuration was returned, whether a correct one was on its slate, and what the surrogate predicted for each — has not been extracted from the run records, and we mark it as the obvious next check.

Additional systems. AutoML4Clust (Tschechlov et al., 2021) is run with silhouette as its supplied index, but over its own search space of four k-center algorithms and with a budget of 100 SMAC loops, neither of which matches CLUSTOPT's space or its 50-evaluation budget. It is therefore a supplementary, unmatched comparison, and it has so far been run on the controlled split only, where it reaches 0.516 mean Best-View ARI. cSmartML, AutoCluster, TPE-AutoClust and CLAMS are not compared because neither offers an implementation we could drive over a fixed, shared configuration space without re-implementing it, which would raise the same fidelity question as above.

R EXTERNAL INDEX RANKING

Table 30 lists all 67 primary indices on the canonical candidate basis, ordered by available-case utility, with coverage, the rank restricted to the 49 fully covered indices, and the number of dataset–view units on which the index appears in the top five and top ten.

The registry is 7 classical, 10 established, 46 newly designed and 4 modern comparators. Utility is *available-case*: it is computed on the units where the index is defined and is never coverage-adjusted, and coverage is reported as a separate column so that the two are not silently combined. Ranking across indices with very different coverage is not meaningful, which is why the full-coverage rank is given alongside. CVDD leads the available-case ordering, but it is *defined* on only 102 of the 150 units — a coverage of 0.68, not a count of units on which it wins — so it does not appear in the full-coverage ranking at all, and its first place is a statement about the subset where it is computable.

Table 31 reports the 46 newly designed indices individually. The group mean is the lowest of the four groups while several individual indices are the single best available choice on particular units; both readings come from this table, and the difference between them is the paper's argument at the level of a single index.

Table 29: Index utility on the independent corpus. Utility is available-case: computed on the units where the index is defined, and never coverage-adjusted. Rank is over all 67 primary indices; FC-rank is over the 49 with complete coverage. The three indices marked • are the generalist core identified on the development repository. The full ranking is in Appendix R.

Index	Group	Utility	Coverage	Rank	FC-rank
`cvdd`	Modern	0.5457	102/150	1	–
• `silhouette`	Original	0.5026	150/150	2	1
• `noise_aware_silhouette`	Established	0.4939	150/150	3	2
• `gridness_fft_acf`	New46	0.4820	150/150	5	4
`cvnn`	Modern	0.4704	148/150	10	–
`dcsi`	Modern	0.3864	145/150	44	–
`cdbw`	Modern	0.3829	147/150	46	–
Group means over all 67 primary indices					
Original (7 indices)		0.4249			
Established (10 indices)		0.4337			
New46 (46 indices)		0.4013			
Modern (4 indices)		0.4463			

Table 29 extracts the headline rows: the three generalist core indices identified internally, the four modern comparators, and the group means. The core's per-source stability — the external half of the transfer claim in Section 7 — is plotted in Figure 3A of the main text.

Table 30: All 67 primary indices on the canonical candidate basis, ordered by available-case utility. Coverage is the number of the 150 dataset–view units on which the index is defined; utility and coverage are separate columns and no coverage-adjusted score is formed.

Rank	Index	Group	Utility	Coverage	FC-rank	Top-5	Top-10
1	`cvdd`	Modern	0.5457	102	–	18	41
2	`silhouette`	Original	0.5026	150	1	31	57
3	`noise_aware_silhouette`	Established	0.4939	150	2	33	51
4	`neighborhood_purity_multi_k`	Established	0.4832	150	3	18	45
5	`gridness_fft_acf`	New46	0.4820	150	4	25	48
6	`avg_pca_isotropy`	Established	0.4793	150	5	24	51
7	`neighborhood_purity`	Established	0.4775	150	6	18	34
8	`min_centroid_distance`	Established	0.4759	150	7	19	42
9	`coggins_jain_index`	Original	0.4712	150	8	19	36
10	`cvnn`	Modern	0.4704	148	–	21	43
11	`direction_entropy`	New46	0.4605	150	9	15	32
12	`blobness_log_dog`	New46	0.4604	149	–	26	42
13	`cluster_size_balance_entropy`	Established	0.4546	150	10	16	30
14	`axial_symmetry`	New46	0.4523	150	11	20	37
15	`arc_circle_fit`	New46	0.4485	150	12	17	37
16	`alpha_shape_compactness`	New46	0.4475	149	–	24	38
17	`contour_graph_connectivity`	New46	0.4454	149	–	12	24
18	`ribbon_thickness`	New46	0.4374	150	13	21	34
19	`dunn_index`	Original	0.4359	150	14	5	17
20	`connected_components_count`	New46	0.4307	144	–	14	27
21	`parallel_bands`	New46	0.4275	150	15	12	21
22	`cop`	Original	0.4253	150	16	3	16
23	`radial_uniformity`	New46	0.4250	150	17	14	26
24	`hough_line_strength`	New46	0.4243	150	18	10	22
25	`ladderness`	New46	0.4208	150	19	12	26
26	`corner_junction_density`	New46	0.4202	149	–	17	29

Rank	Index	Group	Utility	Coverage	FC-rank	Top-5	Top-10
27	lbp_stationarity	New46	0.4183	142	–	14	24
28	min_intercluster_distance	Established	0.4166	150	20	6	14
29	curvature_consistency	New46	0.4145	150	21	10	21
30	gridness_2d	New46	0.4093	150	22	11	23
31	peak_to_background_hough	New46	0.4078	150	23	7	17
32	hollow_score	New46	0.4067	150	24	8	22
33	convexity_ratio_image_based	New46	0.4032	150	25	8	19
34	convexity_ratio	New46	0.4017	150	26	8	16
35	corner_sharpness	New46	0.4013	150	27	5	11
36	morphological_band_count	New46	0.4010	149	–	11	21
37	fractal_dimension	New46	0.3993	144	–	9	25
38	ridge_strength	New46	0.3961	149	–	9	17
39	polygonality	New46	0.3943	150	28	5	11
40	calinski_harabasz	Original	0.3942	150	29	9	18
41	periodicity_1d	New46	0.3895	150	30	4	16
42	hough_arc_circle_strength	New46	0.3892	150	31	13	18
43	glcm_haralick	New46	0.3881	144	–	5	16
44	dcsi	Modern	0.3864	145	–	8	18
45	piecewise_linearity	New46	0.3864	150	32	3	11
46	cdbw	Modern	0.3829	147	–	6	14
47	euler_holes_count	New46	0.3804	142	–	11	15
48	orientation_histogram_entropy	New46	0.3799	150	33	7	16
49	parabolicity	New46	0.3799	150	34	12	19
50	edge_coherence	New46	0.3791	150	35	6	15
51	ellipse_fit_score	New46	0.3785	150	36	5	11
52	rectangularity	New46	0.3758	150	37	9	15
53	connectivity_components	New46	0.3735	150	38	7	16
54	thickness_uniformity	New46	0.3735	150	39	12	18
55	cluster_size_imbalance_entropy	Established	0.3733	150	40	9	11
56	dbcv	Original	0.3732	149	–	5	7
57	davies_bouldin	Original	0.3722	150	41	5	7
58	eccentricity_ratio	New46	0.3661	150	42	5	14
59	line_straightness	New46	0.3648	150	43	5	13
60	skeleton_connectivity	New46	0.3641	150	44	2	12
61	turning_points_count	New46	0.3627	150	45	7	12
62	circularity_compactness	New46	0.3611	150	46	6	11
63	mst_smoothness	New46	0.3557	150	47	3	5
64	bimodality_anti_thickness	New46	0.3552	150	48	5	10
65	s_dbw	Established	0.3446	149	–	1	4
66	avg_within_cluster_dispersion	Established	0.3383	150	49	1	5
67	symmetry_score	New46	0.3214	144	–	4	6

Table 31: The 46 newly designed structural indices, individually. Utility is available-case on the canonical basis; coverage is the number of the 150 dataset–view units on which the index is defined; *rank* is over all 67 primary indices; the last four columns count the units on which the index is ranked first, or appears in the top three, five or ten. The group mean is the lowest of the four groups while several individual indices are the single best available choice on particular units; both readings come from this table.

Index	Utility	FCPS	sk-learn	real	Cov.	Rank	#1	Top-5
gridness_fft_acf	0.4820	0.587	0.541	0.424	150	5	4	25
direction_entropy	0.4605	0.545	0.511	0.412	150	11	3	15
blobness_log_dog	0.4604	0.556	0.522	0.403	149	12	3	26
axial_symmetry	0.4523	0.463	0.566	0.389	150	14	6	21
arc_circle_fit	0.4485	0.529	0.478	0.413	150	15	1	17
alpha_shape_compactness	0.4475	0.521	0.474	0.414	149	16	5	24
contour_graph_connectivity	0.4454	0.452	0.558	0.383	149	17	4	12
ribbon_thickness	0.4374	0.527	0.471	0.397	150	18	2	21
connected_components_count	0.4307	0.494	0.449	0.403	144	20	3	14
parallel_bands	0.4275	0.423	0.520	0.379	150	21	5	12
radial_uniformity	0.4250	0.396	0.498	0.393	150	23	6	14
hough_line_strength	0.4243	0.491	0.502	0.366	150	24	2	10
ladderness	0.4208	0.476	0.454	0.389	150	25	1	12
corner_junction_density	0.4202	0.392	0.429	0.423	149	26	4	17
lbp_stationarity	0.4183	0.365	0.497	0.386	142	27	7	14
curvature_consistency	0.4145	0.452	0.434	0.395	150	29	2	10
gridness_2d	0.4093	0.408	0.424	0.402	150	30	4	11
peak_to_background_hough	0.4078	0.539	0.428	0.364	150	31	1	7
hollow_score	0.4067	0.510	0.425	0.371	150	32	2	8
convexity_ratio_image_based	0.4032	0.384	0.438	0.389	150	33	1	8
convexity_ratio	0.4017	0.505	0.402	0.376	150	34	2	8
corner_sharpness	0.4013	0.472	0.430	0.368	150	35	0	5
morphological_band_count	0.4010	0.428	0.494	0.344	149	36	0	11
fractal_dimension	0.3993	0.379	0.434	0.385	144	37	1	9
ridge_strength	0.3961	0.387	0.444	0.372	149	38	1	9
polygonality	0.3943	0.460	0.450	0.348	150	39	0	5
periodicity_1d	0.3895	0.418	0.439	0.356	150	41	0	4
hough_arc_circle_strength	0.3892	0.388	0.463	0.350	150	42	6	13
glcm_haralick	0.3881	0.418	0.411	0.367	144	43	1	5
piecewise_linearity	0.3864	0.369	0.448	0.358	150	45	0	3
euler_holes_count	0.3804	0.433	0.413	0.347	142	47	2	11
orientation_histogram_entropy	0.3799	0.350	0.384	0.385	150	48	2	7
parabolicity	0.3799	0.375	0.444	0.347	150	49	6	12
edge_coherence	0.3791	0.347	0.380	0.387	150	50	0	6
ellipse_fit_score	0.3785	0.363	0.426	0.357	150	51	1	5
rectangularity	0.3758	0.316	0.390	0.383	150	52	0	10
connectivity_components	0.3735	0.435	0.450	0.317	150	53	0	7
thickness_uniformity	0.3735	0.359	0.388	0.369	150	54	0	12
eccentricity_ratio	0.3661	0.366	0.392	0.352	150	58	0	5
line_straightness	0.3648	0.366	0.392	0.350	150	59	0	5
skeleton_connectivity	0.3641	0.343	0.402	0.349	150	60	0	2
turning_points_count	0.3627	0.340	0.386	0.356	150	61	1	7
circularity_compactness	0.3611	0.319	0.377	0.363	150	62	1	6
mst_smoothness	0.3557	0.356	0.386	0.340	150	63	0	3

Index	Utility	FCPS	sk-learn	real	Cov.	Rank	#1	Top-5
bimodality_anti_thickness	0.3552	0.352	0.366	0.350	150	64	2	5
symmetry_score	0.3214	0.359	0.346	0.297	144	67	2	4

S RUNTIME

Table 32 reports runtime under the accounting rule of Appendix A, with the three scopes kept explicitly separate because they answer different questions and are easy to conflate.

- **Pooled** is the median over all 50×3 dataset–view records. It is the right number for "what does one view cost on a typical unit", and it is *not* a single-view deployment figure, because it pools the cheap single-coordinate views with the expensive joint view.
- **Single view** is the median over the `xy_2d` records only. It is the deployment-style figure and is the scope paired with single-view accuracy.
- **Three-view** is the median over datasets of the summed cost of all three views. It is the scope Best-View ARI requires, because Best-View selects among partitions that only exist if all three views were run.

The two rightmost columns decompose the pooled cost into physical search and arm scoring. That split is the informative one: the extended-inventory AutoClust arms spend roughly 40 s per record on scoring alone, against about 4 s of search, because that system evaluates every candidate under every available index. CLUSTOPT and ML2DAC arms are dominated by search instead. Only these two components carry non-zero values in the persisted records; meta-feature extraction and model initialization are logged as zero and are *not* inferred, so the two columns do not sum to the pooled median.

Figure 12 plots the single-view scope against single-view accuracy, which is the pairing the second bullet above requires; Figure 11 plots the three-view scope. The two scopes are never mixed, and the Pareto frontier is not the same on both.

No controlled runtime figure is reported anywhere in this paper. Controlled runtime records exist but were captured under different timer boundaries from the external ones, and combining incompatible timer scopes would be worse than reporting nothing. All runtime evidence therefore comes from the external benchmark, which has one accounting rule.

Table 32: Runtime by scope. **Pooled** is the median over all 50×3 dataset–view records; **single view** is the median over the `xy_2d` records only; **three-view** is the median over datasets of the summed cost of all three views, which is the scope Best-View ARI requires. The three columns answer different questions and are never mixed. Only physical search and arm scoring carry non-zero values in the persisted records; meta-feature extraction and model initialisation are logged as zero and are not inferred.

Code	Pooled (s)	Single view (s)	Three-view (s)	Search (s)	Scoring (s)
A0	16.23	15.79	48.39	4.42	12.01
A1	17.49	17.03	51.81	4.42	13.09
A2	44.00	43.84	130.11	4.42	39.31
A3	44.87	44.81	133.10	4.42	40.56
C0	12.95	15.27	41.28	11.72	2.36
C1	15.39	19.37	47.22	13.16	2.35
C2	18.05	18.05	66.88	15.67	2.39
C3	14.19	15.25	41.39	11.50	2.37
C4	16.08	20.37	48.33	13.67	2.39
C5	17.01	20.81	62.86	14.57	2.38
M0	6.78	6.84	23.07	5.44	0.86
M1	6.46	6.21	20.23	5.44	0.46
M2	6.30	5.55	21.89	5.44	0.33
M3	6.19	5.83	20.76	5.44	0.23

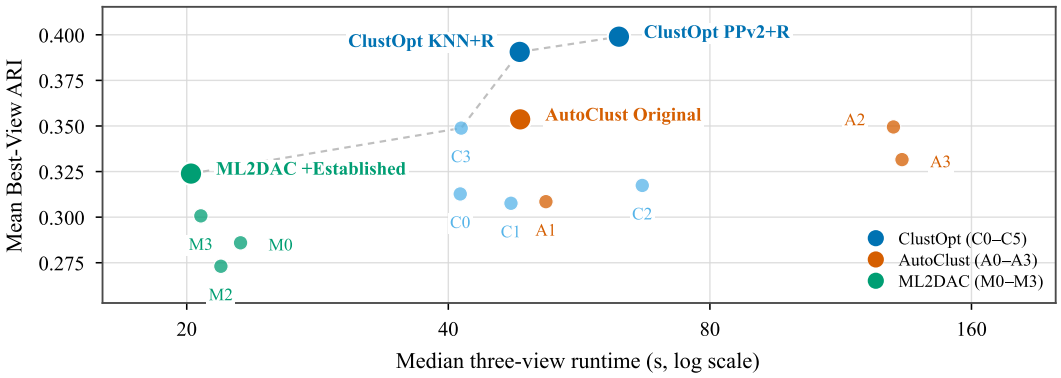

Figure 11: Accuracy against measured cost on the three-view scope, all 14 registered arms: mean Best-View ARI against median three-view runtime (Appendix A); the runtime axis is logarithmic because the arms span 20–133 s. Both CLUSTOPT arms are plotted: C4 is the deployed system throughout this paper and C5 the policy-selector variant, which this figure's labeling still foregrounds. Named arms carry their name and the rest the arm code of Table 27. Runtimes cover search and arm scoring only, excluding meta-feature extraction (Appendix A). Extending AutoClust's inventory moves it right without moving it up. The dashed guide joins the arms flagged Pareto-efficient in the frozen analysis; that refers to the displayed point estimates and is not a claim of statistical dominance.

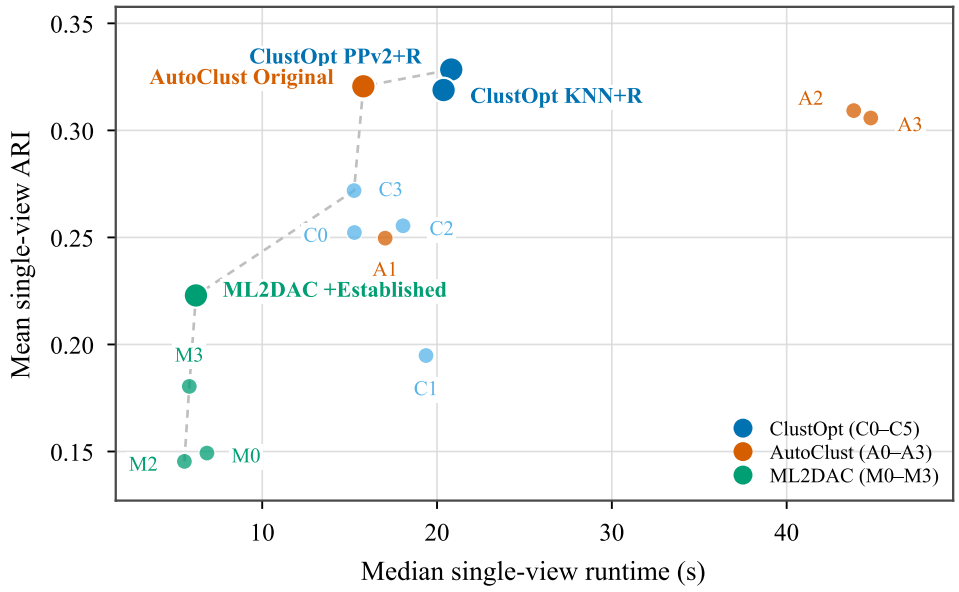

Figure 12: Accuracy against cost on the single-view scope: mean `xy_2d` ARI against median `xy_2d` runtime, all 14 arms. Conventions follow Figure 11, including its labeling of C5 rather than the deployed C4 as the foregrounded CLUSTOPT arm: the four named arms are named, the remaining ten carry the arm code of Table 27, and the dashed guide joins the arms flagged Pareto-efficient in the frozen analysis. AutoClust's best arm becomes Pareto-efficient on this scope, because the three-view figure charges it for views its selection exploits less well.

T CORRECTIONS AND PROVENANCE

Two faults in the external index-evaluation layer were found after external results had been seen. Both are recorded here because their timing matters to how much interpretive freedom they could have created.

Dispatch fault. The evaluation harness ignored the registry field declaring how each index is implemented and applied a float conversion to a structured result object. Six of the 67 indices — the four modern comparators and two image-based indices — therefore produced no values at all on any candidate. The harness was corrected to dispatch on the declared implementation kind, and only those six indices were recomputed, on the unchanged candidate banks.

Candidate-frame fault. The utility computation assembled its candidate frame by adding a row only when some index produced a value, which made the candidate universe depend on index validity. This excluded 392 valid single-cluster partitions from every index's comparison set. Because the frozen utility is frame-relative — its top-k cut-offs and its eligibility guard are fractions of the frame size, and an index's invalid entries are ranked worst rather than dropped (Appendix C) — this shifted $6{,}822$ of $9{,}102$ utility values. The candidate universe is now derived from the candidate bank alone, independent of any index, and all 67 indices were recomputed on that basis.

Baseline index-matrix naming fault. Separately from the two external-layer faults above, the index matrix that feeds the baselines' learned components looked index values up by registry key, whereas the evaluator returns them under each index's internal name. For two image-based indices (`convexity_ratio_image_based`, `hough_arc_circle_strength`) the two names differ, so their columns were empty for every partition. The +New46 and Extended arms of both baselines therefore ran with 51 and 61 indices rather than the intended 53 and 63. CLUSTOPT is unaffected: it reads the same indices under their internal names, and both are live in its vocabulary (Appendix A). The affected arms are reported at their realized inventories; their corrected values would require a rebuilt index matrix and a re-run, which this version does not contain.

What was not affected. Neither correction altered the method outputs, the candidate slates, the candidate ARIs, any index implementation, or the utility definition and its weights. The end-to-end results of Section 7 come from the method-execution layer and are untouched by both. Each fault was diagnosed from a structural invariant rather than from a performance figure, and each repair scope was frozen before corrected values were inspected.

Historical comparisons are excluded. Earlier internal comparisons against AutoClust and ML2DAC were later found to be unpaired, with candidate traces differing between arms in ways the protocol did not control, and the historical ML2DAC scorer returned a non-computable value for 54 of its 61 index classes, leaving the predicted index without influence on selection. Those comparisons are excluded from this paper's claims. Historical controlled ML2DAC values remain visible in Table 18 only as legacy descriptive references, and no inferential or causal claim relies on them (Appendix J). The corrected ML2DAC scorer was produced for Stage 4 and used there, and was subsequently replayed once on the controlled repository to supply the controlled ML2DAC comparison of Section 5.3. Every *authoritative* baseline comparison in this paper — the whole of Section 7 and Appendix P — therefore comes from re-executions under the protocol of Section 4, with the corrected ML2DAC scorer.

Post hoc additions. The external method registry names 14 methods and records that none may be added after outcomes are known. The criterion baselines and additional systems introduced in this version were added after unblinding, at the request of internal review. They are therefore reported separately from the registered arms, excluded from every pre-specified test family, and were run under the unchanged protocol on the unchanged prepared arrays — and on the controlled split only, so the registered external comparison is untouched by them.

Frozen identifiers. The external method registry was frozen before any external execution and records that no method may be added after outcomes are known; it names 14 methods and pins the external submodule commit. The external corpus manifest hash is `709712107424ad4e` and all 50 representation checksums match it. The utility definition, the policy set, the reranker registry and the search protocol each carry their own recorded hash.

U EXTENDED RELATED WORK

Section 2 states the positioning compactly; this appendix gives the full discussion, organized thematically.

Automated clustering, and who already adapts the criterion. Automated clustering inherits the combined algorithm-selection and hyperparameter-optimization formulation from supervised AutoML but without a validation loss, and the systems differ in which part of the pipeline they adapt. It is important to state plainly that adapting the clustering criterion is established prior work and is not what we claim as new. Muravyov & Filchenkov (2017) add a meta-learning step that predicts the cluster-validation index best suited to a dataset's characteristics before an algorithm is selected; this is one of the earliest explicit CVI-selection proposals. cSmartML (ElShawi et al., 2021) performs meta-learning for *algorithm and evaluation-criteria selection* over eight clustering algorithms and then tunes hyperparameters with an evolutionary algorithm against multiple objectives. AutoCluster (Liu et al., 2021) is motivated by precisely the observation we start from — that no general CVI is consistently superior, which its authors name, with the lack of comprehensive meta-features, as a challenge for model optimization in automated clustering — and responds with two components: clustering-oriented meta-feature extraction (a spatial-randomness statistic and landmarkers, alongside traditional statistical and PCA meta-features), and a multi-CVI clustering-ensemble construction that grid-searches the hyperparameters of meta-learned candidate algorithms *under different CVIs* and combines the resulting partitions by majority voting, so as to balance the measuring criteria of different CVIs. Among the systems we discuss, only AutoML4Clust (Tschechlov et al., 2021) leaves the index entirely to the user: it adapts the search, and the internal metric is a required input alongside the dataset and the budget, the authors describing the system as remaining generic with respect to the clustering algorithms, metrics and optimizers used.

Our two baselines, in mechanism. ML2DAC (Treder-Tschechlov et al., 2023) builds a meta-knowledge repository from historical runs and, for a new dataset, retrieves similar datasets by meta-features, *predicts a single CVI* to optimize, and warm-starts and restricts the configuration space, coupling CVI recommendation with Bayesian optimization. AutoClust (Poulakis et al., 2020) should not be reduced to a battery of indices. Its meta-features are themselves internal CVI values, obtained by first running MeanShift on the dataset; algorithm selection is k-NN with majority voting over those meta-features; hyperparameters are tuned by Bayesian optimization with a Tree-structured Parzen Estimator; and the optimization target is a learned quality surrogate, an MLP regressor taking the ten internal CVI values of a candidate partition and predicting that partition's ARI. Both establish that the criterion can be learned, which is the premise we build on.

Where CLUSTOPT actually differs. Not in whether a criterion is learned, but in the granularity and semantics of what is learned, and in what the learned object is used for. A categorical prediction (Muravyov & Filchenkov, 2017; Treder-Tschechlov et al., 2023) yields exactly one criterion per dataset and cannot express that two indices are jointly informative, nor that a third is actively misleading here. A recommended criterion *set* (ElShawi et al., 2021) lifts that restriction, but the set is chosen before the objective is formed and its members enter as separate objectives rather than as weighted contributions to one criterion. An ensemble over CVIs (Liu et al., 2021) likewise uses several indices, but the combination rule is fixed in advance rather than conditioned on the dataset, and it acts by reconciling *partitions* after the fact rather than by forming the objective the search optimizes. A global surrogate (Poulakis et al., 2020) is learned once over all datasets, so what varies per dataset is the algorithm and configuration, not the criterion. CLUSTOPT predicts, for each index in the vocabulary, how well ranking candidates by that index would agree with ranking them by true quality *on this dataset and view* — a continuous profile, not a choice — composes a single scalar search objective from a weighted subset of that profile, and then learns final slate ranking as a separate decision. Whether the added flexibility is realized depends on prediction accuracy, which we report directly rather than assume (Appendix E).

Validity indices. The classical vocabulary is organized around compactness and separation (Rousseeuw, 1987; Caliński & Harabasz, 1974; Davies & Bouldin, 1979), with Halkidi & Vazirgiannis (2001) adding within-cluster scatter and between-cluster density. Density-based indices target different structure — DBCV (Moulavi et al., 2014) scores density connectivity and CDbw (Halkidi & Vazirgiannis, 2008) uses multiple representatives per cluster — and connectivity-oriented measures form a third group: CVNN (Liu et al., 2013) penalizes nearest-neighbor disagreement across bound-

aries, CVDD (Hu & Zhong, 2019) combines a density-involved distance with separation, and DCSI (Gauss et al., 2023) targets separability. All four density- and connectivity-oriented comparators are evaluated in Section 6. Our claim is not that a group dominates but that which group is informative varies by dataset, and we measure that variation rather than asserting which index fails on which geometry.

How validity indices are evaluated, and what is known about when they work. Our utility target is an instance of the evaluation methodology of Vendramin et al. (2010): generate a collection of candidate partitions of a labeled dataset, and score a relative validity criterion by the agreement between the ranking it induces and the ranking induced by an external index. Arbelaitz et al. (2013) compare 30 indices in that spirit across controlled factors and conclude that no index dominates; Gagolewski et al. (2021) reach a more skeptical conclusion about several popular indices against expert labels; and Simpson et al. (2026a) revisit the Vendramin protocol for 26 indices. The closest work to our premise is Simpson et al. (2026b), who apply instance space analysis to nine indices over 18,351 synthetic datasets and eight clustering algorithms to chart how index performance depends on dataset features. That study is analytic: it maps regions of strength. CLUSTOPT is operational: it predicts the per-index utility of an unseen dataset from label-free descriptors, composes a search criterion from the prediction, and is evaluated by the clustering that results. Jaskowiak et al. (2016) build ensembles of relative validity criteria with fixed combination rules and study which members to include; our uniform-weighting arm is the analog of an unselective ensemble, and the dataset-independent profile $\bar{\mathbf{u}}$ of a selective but unconditioned one (Appendix I).

Meta-learning, ranking and deep clustering. Selecting a clustering *algorithm* from meta-features is established (Ferrari & de Castro, 2015; Pimentel & de Carvalho, 2019), including from supervised graph embeddings of the dataset (Cohen-Shapira & Rokach, 2021); zero-shot pipeline selection (Singh et al., 2024) and pipeline ensembles (ElShawi & Sakr, 2022) extend it, and Poulakis et al. (2024) survey the area. It is the unsupervised instance of algorithm selection (Rice, 1976; Kerschke et al., 2019); we instead predict which *criterion* will rank candidates well. Portfolio work reports the virtual best solver as the ceiling selection could reach (Xu et al., 2008). We use that device for four distinct ceilings and keep them separate throughout: the *slate* oracle over one searched candidate slate, a subset oracle over index subsets, the reranked *policy* oracle over the evaluated policy portfolio, and Best-View ARI over the three pre-specified views. Our reranker is closer to learning-to-rank over an already-generated candidate set (Liu, 2009) than to a second search, though the analogy is partial, since our scorer is used only through its $\arg\max$ within a slate and so must identify the slate maximizer rather than the whole ordering. Deep clustering learns a representation and a clustering objective jointly (Xie et al., 2016; Caron et al., 2018), changing the space in which clustering happens; CLUSTOPT operates on fixed representations and addresses label-free model selection over configurations, so the two are complementary. Finally, von Luxburg et al. (2012) argue that clustering comparisons are meaningful only relative to a stated use, which is why we report paired dataset-level statistics rather than differences of means.

V EXTENDED DISCUSSION AND LIMITATIONS

Section 8 states the mechanisms and the limitations compactly; this appendix gives the full account.

The two mechanisms. Adding validity indices is normally reported as neutral or mildly positive, and our uniform arms reproduce that at a fixed budget; the online arm shows why the framing is incomplete, since the same enlargement costs -0.058 ARI while the oracle over the same indices gains $+0.047$. A larger inventory therefore does not move the criterion uniformly in either direction: the added indices help on some datasets and hurt on others, and a system that cannot condition on the dataset cannot tell those cases apart and receives the difference as noise. We describe this as an observed pattern in the two arms above; we did not estimate a variance component and do not claim one. The seeded criterion comparison locates the conditioning gain: every criterion, from uniform weighting to random search to the predicted profile, produces a slate of almost the same quality (oracles within 0.006), and what separates them by up to 0.258 ARI is predominantly which candidate of that slate is returned. Improving $\hat{\mathbf{u}}$ makes the criterion a better proxy for that choice; reranking acts on the same final selection with a learned scorer, using information the criterion does not have. Whether a regime-specific reranker subsumes the conditioning advantage is untested, which is why we do not present the two gains as additive (Appendix V.1). Because $J_{\mathcal{S}}$ is a weighted sum of

internal indices and is not ARI, its $\arg\max$ need not coincide with the slate maximizer; a residual is therefore expected, though we prove no lower bound on it. The empirical signature matches — it improves all 20 controlled policies, most where the objective was worst, and externally helps most where the search is already strong. Its own ceiling on a given slate is the *slate* oracle, 0.6528 for the fixed policy on the earlier unseeded replay, of which reranking recovers 61%. Separately, and at a different level, the reranked *policy* oracle remains approximately 0.040 ARI above PP V2+R; that gap is *policy-selection* headroom across the evaluated policy portfolio, not a residual on any one searched slate.

Specialists are worth having only if something selects them. Aggregate statistics for the 46 new indices are weak, and taken alone would suggest the vocabulary contribution failed. The conditional view contradicts that: 42 subfamily-level specialization relations survive correction, 27 invisible at family level, and several are the single best available choice on particular external units. Both facts are properties of the same objects, and which is visible depends on whether the system can condition. Richness is a property of the vocabulary; value is a property of the pairing between vocabulary and selection.

Scope and limitations. The method and every experiment concern two-dimensional data with between two and five clusters. The 24 image-based indices are defined on a two-dimensional raster and the 22 pattern-based ones project to two dimensions first (Appendix W), so higher-dimensional data are reached only through a projection or embedding chosen by the user, and the search space cannot return more than five clusters wherever a count is required. Within that scope, the evidence does not promote the policy-selector variant over the fixed configuration, and supports a narrower conclusion than the aggregate means suggest: the positive mean difference against AutoClust is carried by two non-convex generator groups, and on the remaining 44 datasets AutoClust performs better on average with an interval excluding zero (Appendix P.1). That includes the 28 real tabular datasets — low-dimensional projections of real tabular data — on which every system's *source-level mean* Best-View ARI is below 0.16. That is a statement about the mean over that source, not about every dataset in it. What the external benchmark establishes is that dataset-conditioned selection is more robust than indiscriminate use of a large inventory, not that the method transfers to arbitrary domains. Best-View ARI is an oracle over three pre-specified views, so it bounds what a perfect view selector would deliver rather than describing a deployable procedure, and external datasets are two-dimensional representations, which keeps the comparison identical across systems but does not test higher-dimensional ambient spaces. With 50 datasets and 40 dependence groups a difference of the size observed between the two CLUSTOPT variants could not be resolved regardless of whether it is real, so claims marked *not established* are unresolved rather than refuted; the controlled replay, though far more powerful, is development-informed and is not an independent test of the finished design. Two defects in the external index-analysis layer were found after external results had been seen and were repaired: both were diagnosed from structural invariants rather than from any performance figure, both repair scopes were frozen before corrected values were inspected, and neither altered the method outputs, the candidate slates, the candidate ARIs, the index implementations or the utility definition, so Section 7 is untouched. Earlier internal baseline comparisons were later found to be unpaired and the historical ML2DAC scorer left its predicted index without influence on selection; those comparisons are excluded. Historical controlled ML2DAC values are still reported as legacy descriptive references; the controlled ML2DAC figure used in the main text comes from a single later replay of the corrected scorer. The controlled criterion baselines were specified after the external outcomes were known, but were evaluated only on Split 1; no post-hoc arm is reported on the external corpus (Appendix T). Several planned studies are absent from this version and their claims with them: a second, natively two-dimensional external corpus; post-hoc external criterion baselines; a multi-seed external repetition; the reranker feature ablations; and the budget, utility-weight and meta-feature-block sensitivity studies. Finally, selection among near-tied candidates can depend on floating-point details, so we pin thread counts rather than claim bit-level reproducibility (Appendices J, T).

Three directions this leaves open. First, the view is still selected by an oracle over the three pre-specified projections, and it could instead be predicted from the same label-free meta-features the utility profile already uses; that would remove the one oracle the headline metric still depends on. Second, the reranker's remaining headroom is directly measurable as the gap to the slate oracle, which makes it an unusually well-posed target: the ceiling is known, the inputs are already computed, and no additional clustering is required to close it. Third, low-dimensional projections of real tabular

data are the regime in which this method has least to offer, since its vocabulary is largely geometric; that is precisely where a vocabulary of non-geometric descriptors would be tested against a clear baseline on which it currently does least well.

V.1 Experiments deferred to a later version

Three studies were specified for this version and not completed. None of the paper's claims depends on them; we record what each would establish.

Does the search criterion matter once the reranker exists?. The controlled evidence establishes a conditioning benefit measured *before* reranking (Table 16) and a reranker benefit measured within the conditioned regime (Table 15). Because criteria differ in which candidate they return rather than in the slate they produce, and a reranker also acts by selecting from a slate, the two gains draw on the same headroom and are not presented as additive. Testing whether the conditioning benefit persists once every criterion receives its own reranker requires one reranker per criterion regime, fitted on development slates produced under that criterion: reusing CLUSTOPT's reranker on another criterion's slate is an out-of-distribution application that would bias the comparison in CLUSTOPT's favor. For CLUSTOPT's own regime those slates exist (48,039 development slates, 1,537,248 candidate rows); for the other criteria they do not, and generating them costs an estimated 77–107 hours per regime. The headroom a $\bar{\mathbf{u}}$ reranker would compete for is visible in Table 17: 0.109 against CLUSTOPT's 0.089. This is the first experiment we would run next.

Reranker feature-group ablations. The deployed reranker uses 347 features in three groups — candidate features, slate-context features and dataset meta-features (Appendix C). Which group carries its gain is not established. The planned ablations withhold one group at a time, replace the FULL60 index values by HEAD14, reduce the model to the candidate's index vector alone (AutoClust's surrogate applied post hoc), and substitute a listwise objective for the pointwise one.

Sensitivity and robustness. Between-seed variation of the external means, the effect of the search budget between 25 and 200 evaluations, the effect of the eight utility-target weights (fixed a priori and never tuned) and the contribution of each meta-feature block were not measured, and we make no robustness claim on their behalf. Every external arm is a single seeded run (Appendix A); the controlled criterion comparison is fully seeded and paired across arms.

W Definitions of the 46 newly designed structural indices

The designed inventory is **46 indices: 22 pattern-based and 24 image-based**, verified against the frozen metric registry (`METRIC_REGISTRY`, 60 entries: 6 core, 8 structural, 22 pattern, 24 image). Every index below is scored on the *full* input space, is oriented so that larger is better, and returns $-\infty$ for a cluster holding fewer than its `min_cluster_points` points so that degenerate clusters can never win a comparison. Partition-level scores aggregate the per-cluster values.

Two shared front-ends. The 22 pattern-based indices act directly on the point coordinates, projecting to two dimensions by PCA when the view carries more. The 24 image-based indices first rasterize the cluster onto a fixed grid through one shared routine (`rasterize_points`) under common parameters (`grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`) and with *partition-level* bounds, so that every cluster of a partition is rasterized into the same coordinate frame and the resulting scores are comparable within a partition.

Two indices that are image-based by design but absent from the internal subgroup matrices. `convexity_ratio_image_based` and `hough_arc_circle_strength` are image-based in the registry and are counted in the 24. Both are live in CLUSTOPT: their utility targets are finite on all 48,039 development records, and they are computed under the internal names `convexity_solidity_image` and `hough_circle_arc_strength`. In the internal Stage-3 matrices they appear under those names and carry no subgroup label, which gives the pattern-versus-image comparison of Appendix L a denominator of $22 + 22 = 44$. The same naming difference caused a lookup defect that left their columns empty in the baselines' index matrix, so that the baselines' realized index counts were 51 and 61 rather than the intended 53 and 63 (Appendix A). Both are fully evaluable on the external corpus (coverage 1.00, available-case utility 0.4032 and 0.3892).

W.1 PATTERN-BASED INDICES (22)

Index	Definition
`alpha_shape_compactness` *Alpha-shape compactness*	Project the cluster to 2D (PCA when the view has more than two coordinates), build an alpha-shape polygon (alpha auto-scaled from point spacing when not supplied) and return its isoperimetric quotient $4\pi A/P^2$, clipped to $[0, 1]$. Parameters: `alpha`, `alpha_scale`, `min_cluster_points`, `eps`. Source: `alpha_shape_compactness.py`.
`arc_circle_fit` *Arc/circle fit*	Fit the minimum enclosing circle; take the radii of all points about its center; score from the relative standard deviation of those radii, clipped at `radial_clip`. Small radial dispersion indicates an arc or ring rather than a filled blob. Parameters: `radial_clip`, `min_cluster_points`, `use_ellipse`, `ellipse_fallback`, `eps`. Source: `arc_circle_fit.py`.
`axial_symmetry` *Axial symmetry*	Take the principal axis u and its normal; bin points along u; within each bin compare the distributions of signed offsets on the two sides of the axis; aggregate the per-bin discrepancies over bins holding at least `min_bin_points` points. Parameters: `min_cluster_points`, `n_bins`, `min_bin_points`, `eps`. Source: `axial_symmetry.py`.
`bimodality_anti_thickness` *Bimodality (anti-thickness)*	Fit the principal axis and take perpendicular offsets d; score the bimodality of the smoothed d histogram from valley depth and peak separation, then blend with an elongation term (`w_elon`) so isotropic blobs cannot score high through incidental bimodality. Parameters: `min_cluster_points`, `bins`, `adaptive_bins`, `min_bins_occupied`, `smooth_kernel`, `min_peak_rel_height`, ... Source: `bimodality_anti_thickness.py`.
`connectivity_components` *Connectivity components (kNN)*	Build a k-nearest-neighbor graph ($k =$`knn_k`) on the 2D cluster, count its connected components c, and return $1/\max(1, c)$. Parameters: `min_cluster_points`, `knn_k`, `eps`. Source: `connectivity_components.py`.
`convexity_ratio` *Convexity ratio (alpha-shape)*	Ratio of the alpha-shape area to the convex-hull area of the cluster, clipped to $[0, 1]$. Concave, hollow or branched shapes score below convex ones. Parameters: `alpha`, `alpha_scale`, `min_cluster_points`, `eps`. Source: `convexity_ratio.py`.
`corner_sharpness` *Corner sharpness*	Compute the convex hull; measure the interior angle at each hull vertex cyclically; score from the incidence of angles sharper than `angle_threshold_deg`. Parameters: `min_cluster_points`, `angle_threshold_deg`, `eps`. Source: `corner_sharpness.py`.
`curvature_consistency` *Curvature consistency*	Order points by the first principal component, estimate signed curvature from consecutive triplets, discard near-straight samples below the 25th percentile of $\lvert\kappa\rvert$, then combine sign consistency (the majority curvature direction, rescaled from $[0.5, 1]$) with low curvature variance and an elongation term. Parameters: `min_cluster_points`, `min_triplets`, `eps`, `w_sign`, `w_var`, `var_scale`, ... Source: `curvature_consistency.py`.
`direction_entropy` *Direction entropy*	Form k-nearest-neighbor edge vectors, weight them by edge length so that long structural edges dominate short noisy ones, and derive a directional-concentration score from the angular vector strength of that weighted distribution, blended with elongation. Parameters: `bins`, `min_cluster_points`, `min_steps`, `step_eps`, `eps`, `w_elon`, ... Source: `direction_entropy.py`.

Index	Definition
`eccentricity_ratio` *Eccentricity ratio*	From the covariance eigenvalues $\lambda_1 \geq \lambda_2$ form $\rho = \sqrt{\lambda_1/\lambda_2}$ and map it to $[0, 1]$ by $1 - \exp(-\rho/r_0)$. Parameters: `min_cluster_points`, `r0`, `eps`. Source: `eccentricity_ratio.py`.
`gridness_2d` *Gridness (occupancy)*	Min–max normalize to the unit square and build an occupancy grid. Rows and columns count as active when their mass exceeds $\max(\texttt{rowcol_presence_thresh}, 1.6 \cdot \text{median} + 1)$, an adaptive threshold that prevents a dense lattice from marking every row active. Requires a minimum number of active rows and columns, then scores the periodicity of the activity profiles by FFT, blended with elongation. Parameters: `grid_size`, `min_cluster_points`, `min_occupied_rows`, `min_occupied_cols`, `rowcol_presence_thresh`, `eps`, ... Source: `gridness_2d.py`.
`hollow_score` *Hollowness*	Build a histogram of normalized radius about the centroid and compare the mass inside the innermost `inner_frac` of the radial range against the outer shell. Parameters: `min_cluster_points`, `n_radial_bins`, `inner_frac`, `eps`. Source: `hollow_score.py`.
`ladderness` *Ladderness*	PCA-rotate the cluster, measure the band strength of the secondary-axis histogram (number of discrete levels K, peak prominence `min_rel_height`, separation `peak_distance_bins`) and multiply by elongation. With $K \leq 1$ the metric falls back to `fallback_weight` times the elongation term. Parameters: `bins`, `min_rel_height`, `peak_distance_bins`, `tau_peaks`, `elong_r0`, `fallback_weight`, ... Source: `ladderness.py`.
`line_straightness` *Line straightness*	Fit a line; with t the along-line and d the perpendicular coordinate, combine an energy term $\text{Var}(t)/(\text{Var}(t) + \text{Var}(d))$ with a thickness term $\exp(-\sigma_d/\sigma_t)$ under weights `w_energy` and `w_thickness`. Parameters: `min_cluster_points`, `eps`, `w_energy`, `w_thickness`. Source: `line_straightness.py`.
`mst_smoothness` *MST smoothness*	Build a k-nearest-neighbor graph and its minimum spanning tree; from the MST degree distribution (branching, scaled by `branch_scale`) and the turning angles along its paths (`w_angle`) score how smooth and unbranched the structure is. A tree that is too sparse or disconnected returns 0. Parameters: `min_cluster_points`, `knn_k`, `w_angle`, `branch_scale`, `eps`. Source: `mst_smoothness.py`.
`parabolicity` *Parabolicity*	Standardize both coordinates, fit a quadratic in both orientations ($y \sim x$ and $x \sim y$), keep the better R^2, and combine it with the normalized quadratic coefficient (scaled by `a_scale`, clipped at `a_clip`) subject to $R^2 \geq$`r2_min`. Parameters: `a_scale`, `a_clip`, `r2_min`, `min_cluster_points`, `eps`. Source: `parabolicity.py`.
`parallel_bands` *Parallel bands*	A partition-level index: each cluster must be straight (at least `straightness_floor`), the cluster orientations must agree in an orientation-invariant sense (`w_angle`), and the band offsets along a shared global normal must be regularly spaced (`w_spacing`). Parameters: `straightness_floor`, `w_angle`, `w_spacing`, `min_cluster_points`, `eps`. Source: `parallel_bands.py`.
`periodicity_1d` *Periodicity (1D)*	PCA-rotate, histogram the secondary-axis coordinate with adaptive bin count, and score spectral periodicity by FFT while ignoring the lowest `ignore_low_freq_frac` of frequencies; blended with elongation. Parameters: `bins`, `adaptive_bins`, `min_bins_occupied`, `ignore_low_freq_frac`, `w_elon`, `elong_r0`, ... Source: `periodicity_1d.py`.

Index	Definition
`piecewise_linearity` *Piecewise linearity*	Sort by the principal coordinate and segment into at most `max_segments` linear pieces by dynamic programming over prefix sums, each segment holding at least `min_seg_points` points and reaching $R^2 \geq$`min_seg_r2`; score from the fit improvement over a single line (scaled by `imp_scale`), blended with elongation. Parameters: `max_segments`, `min_cluster_points`, `min_seg_points`, `max_candidates`, `imp_scale`, `min_seg_r2`, ... Source: `piecewise_linearity.py`.
`radial_uniformity` *Radial uniformity*	Histogram normalized radius about the centroid and score how close the radial density is to that of a filled isotropic cloud, penalizing insufficient mass inside the innermost `inner_frac`. Parameters: `min_cluster_points`, `n_bins`, `inner_frac`, `eps`. Source: `radial_uniformity.py`.
`ribbon_thickness` *Ribbon thickness*	Build the principal-axis basis, bin along the axis, and take each bin's width from the `q_low`–`q_high` quantile range of perpendicular offsets; combine thinness (`w_thin`) with width consistency across bins (`w_cons`) and elongation. Parameters: `min_cluster_points`, `n_bins`, `min_bin_points`, `q_low`, `q_high`, `eps`, ... Source: `ribbon_thickness.py`.
`turning_points_count` *Turning-point count*	Bin along the principal axis with $\mathrm{clip}(\sqrt{n}, 12, 64)$ bins and average the secondary coordinate per bin to obtain a denoised trajectory; count sign changes of its discrete slope, ignoring changes below the `near_zero_percentile` magnitude, and blend with elongation. Parameters: `min_cluster_points`, `min_samples`, `eps`, `near_zero_percentile`, `w_elon`, `elong_r0`. Source: `turning_points_count.py`.

W.2 IMAGE-BASED INDICES (24)

Index	Definition
`blobness_log_dog` *Blobness (LoG/DoG)*	Multi-scale difference-of-Gaussian response over `sigmas`; take the maximum absolute response and normalize by the number of occupied pixels. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `sigmas`, ... Source: `blobness_log_dog.py`.
`circularity_compactness` *Circularity/compactness*	Extract the largest contour of the binarized raster and return its circularity $4\pi A/P^2$. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `eps`. Source: `circularity_compactness.py`.
`connected_components_count` *Connected-component count*	Threshold at $\max(1, 0.30 \cdot \mathrm{peak})$ so that soft Gaussian tails do not fragment into speckle, close gaps morphologically, count connected components and map the count into $[0, 1]$ (decay `a_comp`). Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `a_comp`, ... Source: `connected_components_count.py`.
`contour_graph_connectivity` *Contour-graph connectivity*	Canny edge map (`canny_low`/`canny_high`), requiring at least `min_edges` edge pixels; close small breaks with a 3×3 elliptical kernel so one contour does not appear as many fragments; score the connectivity of the resulting edge graph. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `contour_graph_connectivity.py`.

Index	Definition
`convexity_ratio_image_based` *Convexity ratio (image-based)*	Largest contour of the binarized raster; return the solidity, the contour area divided by the area of its convex hull. The raster counterpart of `convexity_ratio`. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `eps`. Source: `convexity_ratio_image_based.py`.
`corner_junction_density` *Corner/junction density*	Canny edges, then Shi–Tomasi corner detection (`max_corners`, `quality_level`) on the edge image; score the density of detected corners relative to the edge extent. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `corner_junction_density.py`.
`edge_coherence` *Edge coherence*	Sobel gradients; retain gradients above the `mag_thr_quantile` magnitude quantile; score the structure-tensor coherence of their orientations at smoothing scale `smooth_sigma`. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `smooth_sigma`, ... Source: `edge_coherence.py`.
`ellipse_fit_score` *Ellipse-fit score*	Largest contour with at least five points; fit an ellipse by least squares and score the agreement between contour and fitted ellipse. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `ellipse_fit_score.py`.
`euler_holes_count` *Euler hole count*	Binarize, then close heavily (7×7 ellipse, at least two iterations) so that rasterization pinholes are removed and the count reflects genuine topology; enumerate contour hierarchy and count holes above an area threshold. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `close_iters`. Source: `euler_holes_count.py`.
`fractal_dimension` *Fractal dimension (box count)*	Canny edges (at least 40 pixels), then box counting over power-of-two box sizes; the score is the least-squares slope of $\log N(s)$ against $\log(1/s)$. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `fractal_dimension.py`.
`glcm_haralick` *GLCM Haralick features*	Quantize the raster to `levels` gray levels, form gray-level co-occurrence matrices over the configured offsets, and average contrast, homogeneity and entropy across them. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `levels`, ... Source: `glcm_haralick.py`.
`gridness_fft_acf` *Gridness (FFT/ACF)*	Two-dimensional autocorrelation of the raster; keep the `peak_k` strongest peaks subject to a suppression radius and a floor of `min_peak_val`; with at least two peaks, score the regularity of the lattice vectors they define. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `peak_k`, ... Source: `gridness_fft_acf.py`.
`hough_arc_circle_strength` *Hough arc/circle strength*	Canny edges (thresholds auto-selected when unset), then a circular Hough transform (`dp`, `minDist_frac`, `param1`, `param2`, radius bounds as fractions of the image). A detected circle must be supported by the data: the point inlier fraction within `support_tol` must reach `min_point_inlier`, angular coverage over `angle_bins` must reach `min_angle_coverage`, and edge recall must reach `min_edge_recall`. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `hough_arc_circle_strength.py`.

Index	Definition
`hough_line_strength` *Hough line strength*	Canny edges, then the probabilistic Hough line transform (`rho`, `theta`, `threshold`, `min_line_length`, `max_line_gap`); score total detected line support relative to the edge extent. Parameters: `min_cluster_points`, `raster`, `canny_low`, `canny_high`, `rho`, `theta`, ... Source: `hough_line_strength.py`.
`lbp_stationarity` *LBP stationarity*	Normalize to 8-bit and require a foreground fraction of at least `min_foreground_frac`; compute the 8-neighbor local binary pattern and score how stationary its histogram is across image regions. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `tiles`, ... Source: `lbp_stationarity.py`.
`morphological_band_count` *Morphological band count*	Binarize, estimate the dominant foreground orientation and rotate to align it with the horizontal axis; open with a horizontal line kernel of length `line_kernel_frac` of the width; smooth the projection profile (`profile_smooth_k`) and count peaks subject to `peak_min_dist` and `peak_min_prom`. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `line_kernel_frac`, ... Source: `morphological_band_count.py`.
`orientation_histogram_entropy` *Orientation-histogram entropy*	Sobel gradients; retain magnitudes above the `mag_thr_quantile` quantile; score the entropy of the resulting gradient-orientation histogram. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `n_bins`, ... Source: `orientation_histogram_entropy.py`.
`peak_to_background_hough` *Peak-to-background (Hough)*	Canny edges, then probabilistic Hough lines with minimum length and maximum gap set as fractions (`min_line_len_frac`, `max_line_gap_frac`) of the smaller image dimension; score the ratio of accumulated line support to background edge mass. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `canny_low`, ... Source: `peak_to_background_line_ratio.py`.
`polygonality` *Polygonality*	Largest contour; Douglas–Peucker polygonal approximation; score from the resulting vertex count relative to `max_vertices` together with the fidelity of the approximation. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `approx_eps_frac`, ... Source: `polygonality.py`.
`rectangularity` *Rectangularity*	Largest contour; ratio of its area to the area of its minimum-area rotated bounding rectangle. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `eps`. Source: `rectangularity.py`.
`ridge_strength` *Ridge strength (vesselness)*	Frangi-like multi-scale vesselness over `sigmas` with parameters `beta` and `c`; score the fraction of pixels in the `top_quantile` of vesselness response, which favors extended tubular structure over locally bright blobs. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `sigmas`, ... Source: `ridge_strength.py`.
`skeleton_connectivity` *Skeleton connectivity*	Binarize, close (`close_ksize`, `close_iter`) and skeletonize; from the component count, endpoint count, junction count and skeleton coverage form $\exp(-a(\text{comp}-1))\cdot\exp(-b\max(0,\text{end}-2))\cdot\exp(-c\,\text{junc})\cdot(1-\exp(-8\,\text{cov}))$ with $a=$`a_comp`, $b=$`b_end`, $c=$`c_junc`. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `thinning_max_iter`, ... Source: `skeleton_connectivity.py`.

Index	Definition
`symmetry_score` *Symmetry score*	Agreement between the raster and its vertical flip, and between the raster and its horizontal flip; the score is the larger of the two. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`. Source: `symmetry_score.py`.
`thickness_uniformity` *Thickness uniformity*	Euclidean distance transform of the mask and its skeleton; local thickness is twice the distance transform sampled on skeleton pixels (at least `min_skeleton_pixels`); score the uniformity of that thickness distribution. Parameters: `min_cluster_points`, `grid_size`, `raster_mode`, `blur_ksize`, `dilate_iters`, `thinning_max_iter`, ... Source: `thickness_uniformity.py`.