COCLUSTERPFN: AMORTIZED POSTERIOR SIMILARITY FOR TABULAR BAYESIAN CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Most clustering methods return one grouping without revealing which assignments are uncertain. Bayesian methods represent this uncertainty but typically require expensive inference. We introduce CoClusterPFN, a Prior-data Fitted Network (PFN) for Bayesian clustering of tabular data. It is trained on synthetic clustering tasks to predict pairwise posterior co-clustering probabilities in a single forward pass. The resulting posterior similarity matrix (PSM) avoids the label-switching ambiguity of cluster assignments. When the number of clusters is unspecified, the prediction marginalizes over cluster-count uncertainty. The same PSM provides an interpretable summary and can be decoded into hard partitions using decision rules targeting different metrics or selecting different numbers of clusters. Under a Gaussian-mixture prior, CoClusterPFN closely matches PSMs estimated by Markov chain Monte Carlo (MCMC) while being orders of magnitude faster. On 60 CLUBench datasets, CoClusterPFN achieves the best average rank for adjusted Rand index (ARI), normalized mutual information (NMI), and Binder loss, while ranking second for variation of information (VI), whether the cluster count is supplied or unknown. Beyond benchmark performance, an Iris case study demonstrates how CoClusterPFN can help users choose among alternative clusterings. Together, these results support CoClusterPFN as an effective approach to uncertainty-aware clustering.

1 INTRODUCTION

In supervised learning, models need to predict a clearly defined target. In clustering, the goal is instead to group observations. A dataset may support multiple plausible partitions, and even the number of clusters may be unknown or not clearly defined. This ambiguity makes clustering inherently subjective. However, most algorithms return a single clustering, leaving uncertainty hidden.

Bayesian clustering aims to capture this uncertainty, but it can be difficult to apply in practice. A Bayesian model starts with a *prior*, which describes a possible data-generating process. Bayes' rule combines the prior with the observed data and produces a *posterior*. For clustering, the full posterior is a distribution over all possible clusterings, also called partitions (Wade, 2023). Computing this posterior is generally intractable because it is defined over a combinatorially large space. Markov chain Monte Carlo (MCMC) approximates posterior quantities using samples, whereas variational inference optimizes a tractable approximation to the posterior (Andrieu et al., 2003; Blei et al., 2017). Both solve a new inference problem for each dataset, which is typically computationally expensive.

Prior-data Fitted Networks (PFNs) learn this inference step from synthetic data (Müller et al., 2022). A PFN trains once on millions of datasets drawn from a prior. The training task teaches the neural network, typically a transformer, to map an observed dataset directly to a chosen Bayesian posterior quantity. After pretraining, inference on a new dataset only requires a single forward pass. This amortizes the posterior computation across future datasets, moving the expensive part to pretraining.

We introduce CoClusterPFN, a PFN that uses amortized neural inference to approximate the *posterior similarity matrix* (PSM) (Binder, 1978; Fritsch & Ickstadt, 2009). Given N observations, the PSM is an $N \times N$ matrix whose entries give the posterior probability that each pair of observations belongs to the same cluster, conditional on the observed dataset. See Figure 1. Predicting it directly offers advantages over the cluster-assignment outputs of existing PFN methods, such as TabClustPFN (Zhao et al., 2026b) and ClusterPFN (Bhaskaran & Viering, 2025).

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.

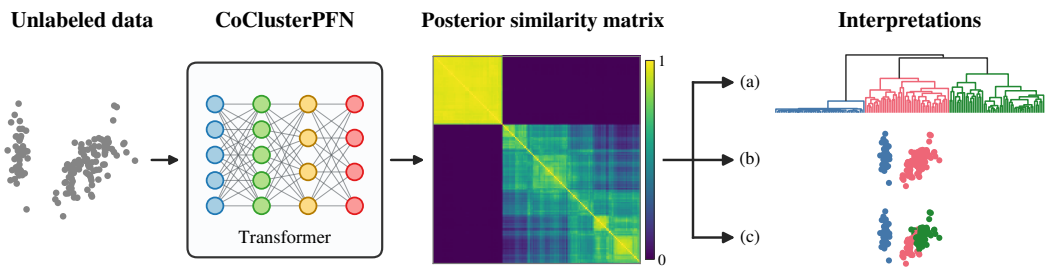


Figure 1: Overview of CoClusterPFN on Iris with the cluster count unspecified. CoClusterPFN predicts a posterior similarity matrix (PSM) through amortized inference, where entry $\widehat{S}_{ij}$ is the predicted probability that observations i and j belong to the same cluster. The same PSM supports (a) a complete-linkage hierarchy and the alternative clusterings shown in panels (b) and (c).

The PSM avoids the ambiguity of arbitrary cluster labels associated with label switching in Bayesian clustering (Stephens, 2000). For example, the labelings $(1, 1, 2, 2)$ and $(7, 7, 3, 3)$ describe the same partition despite using different cluster names. When training a PFN to predict cluster assignments, these equivalent labelings must be handled explicitly. The PSM avoids this issue because it is invariant to cluster relabeling. It also provides a label-invariant training target: the model can be supervised on whether each pair belongs together without aligning cluster-specific output slots.

CoClusterPFN can predict co-clustering probabilities without committing to a particular number of clusters. When the number of clusters is unspecified, the predicted PSM marginalizes over cluster-count uncertainty. This preserves uncertainty arising from alternative cluster counts, which would be lost by conditioning on a single count. Alternatively, when the number of clusters is supplied, CoClusterPFN can predict the PSM conditional on that information.

Beyond this, the PSM aids interpretation. Blocks of high co-clustering probabilities reveal groups of observations likely to belong together, while intermediate probabilities highlight uncertain pairwise relationships (see Figure 1). Users can therefore inspect which pairs are confident or ambiguous.

Because CoClusterPFN returns a PSM rather than a single partition, different decision rules can be applied to the same output, as illustrated in Figure 1. Bayesian decision rules can select partitions by minimizing a posterior expected loss, such as Binder loss (Binder, 1978), while hierarchical clustering provides a faster heuristic alternative. Users can therefore explore partitions selected according to different criteria without rerunning CoClusterPFN.

Our contributions are:

- We develop CoClusterPFN, a PFN that approximates posterior co-clustering probabilities, with support for both supplied and unknown cluster counts.
- We validate the predicted PSM against MCMC and compare CoClusterPFN with amortized clustering methods in challenging 2D Gaussian-mixture experiments.
- We compare CoClusterPFN's direct PSM supervision with responsibility prediction in a controlled ablation and find higher decoded ARI on matched synthetic tasks and CLUBench.
- We evaluate clustering quality across four metrics on 60 CLUBench datasets, and we illustrate how CoClusterPFN can aid interpretation in a case study on Iris.

2 Background and Related Work

We review Bayesian clustering, posterior similarity matrices and their decision rules, Prior-data Fitted Networks (PFNs), and related amortized clustering methods.

Clustering. Let $X = (x_1, \ldots, x_N)$ denote N observations and let $z = (z_1, \ldots, z_N)$ define a partition into K clusters. Cluster labels are arbitrary: relabeling groups does not change the partition. Some clustering methods take K as input, whereas others infer K from data.

Bayesian clustering. Bayesian clustering represents uncertainty through a posterior distribution $p(z \mid X)$ over partitions. Model-based approaches specify a likelihood and priors on unknown quantities; common examples include finite mixture models, mixtures with a prior on the number of components, and Bayesian nonparametric mixtures (Fraley & Raftery, 2002; Miller & Harrison, 2018; Wade, 2023). Loss-based approaches instead construct a generalized Bayesian posterior from a clustering loss (Rigon et al., 2023). Given either form of posterior, Bayesian decision theory can select a single partition by minimizing posterior expected loss (Binder, 1978; Lau & Green, 2007).

Posterior similarity matrix (PSM). Given a posterior distribution $p(z \mid X)$ over partitions, the PSM contains the pairwise co-clustering probabilities (Binder, 1978; Fritsch & Ickstadt, 2009):

$$S_{ij} = \mathbb{E}_{p(z|X)}[\mathbf{1}\{z_i = z_j\}] = P(z_i = z_j \mid X). \tag{1}$$

The PSM is symmetric, has a unit diagonal, and is invariant to permutations of the cluster labels. If the number of clusters is uncertain, S_{ij} averages over that uncertainty. It is a lossy summary: different posterior distributions over partitions can have the same pairwise marginals.

Given retained posterior samples $z^{(1)}, \ldots, z^{(M)}$ obtained by MCMC, the PSM can be estimated using empirical co-clustering frequencies (Fritsch & Ickstadt, 2009):

$$S_{ij}^{\mathrm{MCMC}} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}\left\{ z_i^{(m)} = z_j^{(m)} \right\}. \tag{2}$$

This estimator requires no alignment of cluster labels across samples and allows the sampled partitions to contain different numbers of clusters. Variational inference can also approximate S_{ij}. Under a mean-field factorization (Blei et al., 2017),

$$p(z \mid X) \approx q(z) = \prod_{i=1}^{N} q_i(z_i), \qquad r_{ik} := q_i(z_i = k), \qquad \sum_{k=1}^{K} r_{ik} = 1, \tag{3}$$

where r_{ik} is the responsibility, or marginal probability that observation i belongs to cluster k. Under this factorization, the implied off-diagonal co-clustering probabilities are

$$\widehat{S}_{ij}^{R} = r_i^{\top} r_j = \sum_{k=1}^{K} r_{ik} r_{jk}, \qquad i \neq j. \tag{4}$$

Writing $R = (r_{ik})$, these are the off-diagonal entries of $RR^{\top}$; the diagonal entries are set to one. The same construction can be applied to responsibilities returned by other methods, including responsibility-based PFNs. In each case, it factorizes pairwise probabilities through pointwise responsibilities and therefore cannot represent posterior dependence between assignments. CoClusterPFN does not impose this factorization.

Decision rules from a PSM. Several decision rules can decode a PSM into hard partitions. Binder loss penalizes false pairwise separations and groupings, and its posterior expectation depends only on the PSM (Binder, 1978; Lau & Green, 2007). Selecting pairs independently need not produce a valid partition, and thus requires search over candidate partitions (Lau & Green, 2007; Wade & Ghahramani, 2018). Other criteria include variation of information (VI) and posterior expected adjusted Rand index (ARI) (Fritsch & Ickstadt, 2009; Wade & Ghahramani, 2018). Although the PSM alone does not determine their exact posterior expectations, it supports MinVI-LB, which minimizes a lower bound on expected VI, and MaxPEAR, which approximates expected ARI. Hierarchical clustering of the dissimilarities $1 - S_{ij}$ provides a faster heuristic alternative. We evaluate these approaches.

Prior-data Fitted Networks. PFNs amortize posterior predictive inference by training neural networks on synthetic tasks sampled from a prior. With sufficient model capacity, the population optimum of the PFN log loss is the Bayesian posterior predictive distribution induced by that prior (Müller et al., 2022). PFNs are conditional neural processes (Garnelo et al., 2018) and are commonly implemented using Transformers. At test time, tabular foundation models such as TabPFN and TabICL require only neural network evaluation and can compete with methods that perform expensive hyperparameter tuning separately for each dataset (Hollmann et al., 2025; Qu et al., 2025). PFNs can also condition their predictions on auxiliary information (Chang et al., 2025); the same population-optimum result holds when this information is sampled jointly with each task (Rakotoarison et al., 2026). CoClusterPFN adopts the PFN framework for clustering and, when available, conditions its pairwise predictions on the supplied cluster count.

Amortized clustering methods. Other amortized methods exist, but focus on different outputs. Set Transformer modules have been used to estimate Gaussian-mixture parameters for a fixed number of components (Lee et al., 2019a). Neural Clustering Processes produce approximate posterior samples of assignments sequentially (Pakman et al., 2020), while Deep Amortized Clustering and Cluster-wise Amortized Mixing Coupling Processes construct clusters one at a time and can support an unknown number of groups (Lee et al., 2019b; Liu & Jing, 2022). DeepDPM also infers a variable number of clusters through split–merge operations, but optimizes separately for each dataset rather than learning a cross-dataset posterior inference rule (Ronen et al., 2022). None of these approaches directly predicts a PSM. A related PFN uses a diffusion-transformer decoder trained with flow matching to generate posterior samples of Gaussian-mixture parameters, but its clustering experiments assume a fixed component count and do not return assignments or a PSM (Reuter et al., 2025). ZEUS learns transferable zero-shot embeddings for downstream clustering (Marszałek et al., 2025), while TableDC jointly learns a tabular representation and cluster assignments using a covariance-aware similarity (Rauf et al., 2025). TagCC incorporates semantic information through LLM-derived textual anchors (Zhao et al., 2026a). Other TabPFN approaches refine pseudo-labels initialized by conventional clustering (Li et al., 2025).

Closest predecessors. Attention-Based Clustering (ABC) is a close architectural predecessor that uses self-attention and binary cross-entropy to learn pairwise same-cluster probabilities for downstream clustering (Coward et al., 2020). Among PFNs, ClusterPFN estimates a posterior over cluster count and predicts responsibilities conditional on a selected count, using two forward passes when K is unknown and a deterministic ordering of cluster labels to handle label switching (Bhaskaran & Viering, 2025). TabClustPFN combines partition and cardinality inference networks and trains its partition output using SoftARI, a differentiable label-invariant relaxation of ARI (Zhao et al., 2026b). SoftARI optimizes partition agreement instead of posterior probabilities.

CoClusterPFN combines direct pairwise prediction with the PFN framework. Unlike responsibility-based PFNs, it does not factorize co-clustering probabilities through cluster-specific assignments, and the same output supports multiple hard-partition decision rules without retraining.

3 CoClusterPFN

CoClusterPFN maps an unlabeled dataset to a predicted posterior similarity matrix (PSM). It is trained on synthetic clustering tasks following the PFN framework. We describe the training prior, prediction target, architecture, and PSM decoding procedure below.

Tabular clustering prior. There is no single definition of a cluster, so the training prior inevitably reflects a modeling choice. We focus on tabular datasets that may contain both numerical and categorical features and design the prior to cover a wide variety of clustering problems within that setting. The prior randomizes the dataset size, dimensionality, number and balance of clusters, and cluster separation. Clusters may be unimodal or multimodal and well separated or overlapping; their components may be Gaussian or Student-t.

For most tasks, latent observations are mapped to observed features using random linear projections and nonlinear transformations. The prior also introduces features unrelated to cluster membership and noisy duplicates of informative features, and generates continuous, binary, nominal, ordinal, and count-valued columns. Occasional missing entries and random reassignment of cluster labels provide additional perturbations. The prior also includes clean Gaussian-mixture tasks to cover conventional continuous clustering problems. We specified the prior settings manually and did not optimize them on CLUBench. Appendix A describes the generative mechanisms and main numerical settings.

Training target. Let $X = (x_1, \ldots, x_N)$ denote a dataset and let $z = (z_1, \ldots, z_N)$ denote its synthetic ground-truth partition into K clusters. We define the pairwise co-membership target $Y_{ij} = \mathbf{1}\{z_i = z_j\}$. CoClusterPFN predicts a posterior similarity matrix $\widehat{S} \in [0,1]^{N \times N}$ from X, either with K supplied or without it. We call these the known-K and unknown-K prediction modes, respectively. We train its entries using binary cross-entropy (BCE) against the corresponding targets Y_{ij}. BCE is a strictly proper scoring rule: its population minimizer is $P(z_i = z_j \mid X, K)$ in the known-K mode and $P(z_i = z_j \mid X)$ in the unknown-K mode.

Architecture. The architecture has three stages:

$$X \longrightarrow (e_1, \ldots, e_N) \longrightarrow (h_1, \ldots, h_N) \longrightarrow \{\widehat{S}_{ij}\}_{i,j=1}^{N}.$$

A pretrained tabular encoder first produces a fixed-width representation e_i for each observation. A Transformer encoder then contextualizes these representations across the dataset, producing h_i. Finally, a shared pairwise multilayer perceptron (MLP) predicts each PSM entry.

Row encoder. We initialize the row encoder from the pretrained TabICL model (Qu et al., 2025). We use its column-embedding and row-interaction modules but omit its supervised in-context prediction stage, so no labels are provided. The column-embedding module represents each feature using its values across observations, while the row-interaction module combines information across features within each observation. Together, they produce fixed-width row embeddings, allowing the same model to process datasets with different numbers of input features.

Dataset-level Transformer. A learned projection followed by a four-layer Transformer encoder maps the row embeddings $(e_1, \ldots, e_N)$ to contextualized states $(h_1, \ldots, h_N)$. Full self-attention allows every observation to attend to all other observations in the dataset (Vaswani et al., 2017). We use no row-position encodings, making the model permutation equivariant: permuting the input rows permutes both axes of the predicted PSM in the same way (Lee et al., 2019a).

PSM prediction via MLP. We set $C := K$ in the known-K mode and $C = 0$ in the unknown-K mode. For each pair (i, j), we form $r_{ij} = [h_i + h_j,\ |h_i - h_j|,\ h_i \odot h_j,\ q(C)]$, where $\odot$ denotes elementwise multiplication and $q(C)$ is a one-hot encoding. A shared MLP g_ϕ predicts $\widehat{S}_{ij} = \sigma(g_\phi(r_{ij}))$, where σ is the sigmoid function. Because $r_{ij} = r_{ji}$, the prediction satisfies $\widehat{S}_{ij} = \widehat{S}_{ji}$ by construction. During training, we evaluate each sampled pair in both prediction modes and average the resulting binary-cross-entropy losses.

At inference, CoClusterPFN encodes the dataset once and evaluates the pairwise head in chunks to construct the complete PSM, with its diagonal set to one. The final PSM and pairwise computation scale as $O(N^2)$, while chunking reduces intermediate memory use. During training, only sampled off-diagonal pairs contribute to the loss. Appendices B.1 and B.2 provide further training details.

Decoding the PSM into a hard clustering. To obtain hard cluster labels, we construct average- and complete-linkage hierarchies from the dissimilarities $1 - \widehat{S}_{ij}$. When K is supplied, each hierarchy is cut into K groups. Otherwise, we consider cuts with $K \in \{2, \ldots, 10\}$.

We score these candidate partitions using three PSM-based decision criteria: MaxPEAR, which approximates posterior expected adjusted Rand index (Fritsch & Ickstadt, 2009); MinVI-LB, based on a lower bound on posterior expected variation of information (Wade & Ghahramani, 2018); and posterior expected Binder loss with equal false-merge and false-split costs (Binder, 1978). Because the search is restricted to hierarchical cuts, it need not find the global optimum over all partitions.

We select a decision rule per metric on 256 independent synthetic validation tasks. MaxPEAR is selected for ARI in both settings and for NMI when K is unknown. MinVI-LB is selected for VI in both settings and for NMI when K is supplied, while MinBinder is selected for N-invariant Binder loss. Appendices B.5 and C.1 define the decoding objectives and evaluation metrics, respectively.

4 Experiments

We compare CoClusterPFN with MCMC and amortized clustering methods on challenging 2D Gaussian mixture model (GMM) data. We compare against a baseline that uses responsibilities to validate our design, conduct a benchmark on CLUBench, and present a case study.

4.1 Matched-prior evaluation

We perform two experiments with synthetic tasks drawn from a two-dimensional Normal–Inverse-Wishart Gaussian-mixture family with $K \in \{2, \ldots, 10\}$. The amortized methods (and our method) are retrained for this prior specifically. The prior imposes no minimum separation between components and can produce substantial overlap and thus challenging clustering problems.

Agreement with MCMC. The conjugate Normal–Inverse-Wishart (NIW) model permits collapsed Gibbs sampling with the mixture parameters integrated out, which we use as a high-budget Monte Carlo reference. We generate 54 datasets, balanced across $K \in \{2, \ldots, 10\}$, and report results on the 45 that pass the prespecified convergence diagnostics in Appendix F. Agreement between the PSMs is close: mean absolute error (MAE) is 0.019 in the known-K mode and 0.030 in the unknown-K mode, while the corresponding correlations are 0.98 and 0.97. In a separate, unoptimized fixed-K timing experiment, warmed-up CoClusterPFN inference was approximately $50{,}000\times$ faster than four sequential CPU MCMC chains.

Comparison with amortized clustering methods. We next compare against Neural Clustering Processes (NCP) (Pakman et al., 2020) and Deep Amortized Clustering (DAC) (Lee et al., 2019b). We do not include them in the CLUBench comparison because their published formulations amortize inference for a specific, fixed-dimensional task distribution.

We retrain all methods on the same NIW–GMM task distribution using method-specific optimization protocols: 30,000 updates for CoClusterPFN, 10,000 for NCP, and 20,000 for each DAC variant. We evaluate the models on 2,500 fresh datasets from the same family, with K withheld and N ranging from 10 to 200. CoClusterPFN predicts a PSM, while NCP constructs an empirical PSM from ten sampled autoregressive partitions. CoClusterPFN and NCP use metric-specific decoders selected on held-out tasks from the same NIW–GMM prior; DAC uses its two published filtering objectives without an additional decoder. The decoder-selection protocol and rules appear in Appendix G.

Table 1: Metric-aligned unknown-count clustering on 2,500 fresh tasks from the shared NIW–GMM family. CoClusterPFN and NCP decoders are selected separately for each metric on held-out synthetic tasks; their count diagnostics use the ARI-selected partitions. Binder is the N-invariant Binder loss.

Method	ARI ↑	NMI ↑	Binder ↓	VI ↓	Exact K ↑	K MAE ↓
CoClusterPFN	**0.189**	0.413	**0.283**	**1.357**	**0.155**	**2.30**
NCP	0.134	0.350	0.306	1.391	0.136	2.33
DAC, anchored filtering	0.117	**0.474**	0.285	1.880	0.024	17.77
DAC, minimum-loss filtering	0.020	0.034	0.698	1.411	0.029	4.30

Table 1 shows that CoClusterPFN obtains the highest mean ARI, lowest Binder loss and VI, highest exact-K accuracy, and lowest K MAE. The paired 95% bootstrap intervals for its ARI advantage over each alternative exclude zero. Anchored DAC obtains the highest NMI but strongly overpartitions the data, as reflected by its much larger K MAE. Complete training protocols, paired intervals, and selected observation-count diagnostics appear in Appendix G.

4.2 Direct PSM supervision versus responsibility prediction

PFNs can be supervised through cluster assignments or through pairwise co-membership targets. We develop a Responsibility-PFN, which predicts assignment probabilities over K output slots and requires Hungarian matching; its implied PSM is obtained using Equation 4. CoClusterPFN instead predicts the co-clustering probabilities directly. Direct PSM supervision is label invariant and exposes each pairwise relationship directly to the loss. We therefore test whether this parameterization supports better hard-clustering decisions.

We compare the primary CoClusterPFN with the Responsibility-PFN that shares its backbone, tabular training prior, optimization budget, and training seed. Both models receive the reference K. We evaluate them on 300 held-out tasks from the matched tabular prior and on the 60 compatible CLUBench datasets (see selection criteria in Section 4.3). For each predicted PSM, we obtain partitions using average linkage, MinBinder, and MaxPEAR.

As shown in Figure 2, direct PSM prediction improves decoded ARI under all three decision rules on both the synthetic tasks and the 60 supplied-K CLUBench datasets. The paired 95% bootstrap intervals exclude zero for all six comparisons. Taking the Responsibility-PFN assignments directly by argmax yields similar performance. This indicates that the advantage is not introduced by constructing and decoding $RR^\top$. On the synthetic tasks, direct PSM prediction also improves BCE, Brier score, and MAE (Table 8, Appendix C.2).

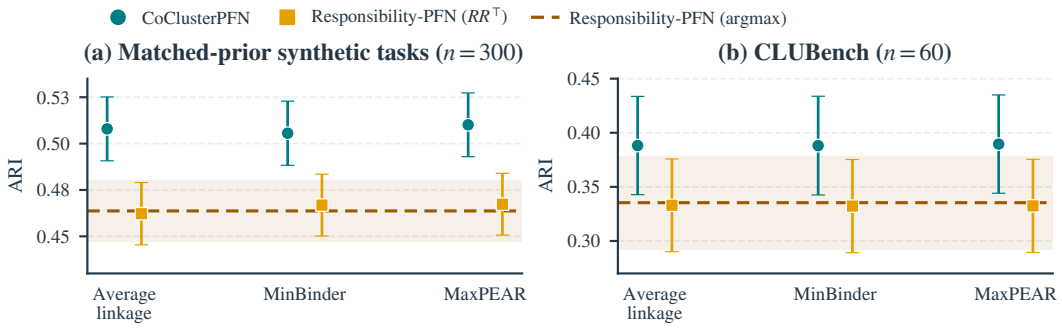


Figure 2: ARI comparison with the reference K supplied. Circles show CoClusterPFN, while squares show the Responsibility-PFN after constructing a PSM as $RR^\top$. Points are means and error bars are standard errors across 300 held-out synthetic tasks or 60 CLUBench datasets. The dashed line shows the mean ARI obtained by assigning each observation to its largest predicted responsibility, and the shaded region shows its standard error.

Table 2: Results on 60 CLUBench datasets with supplied K, reported as mean score (mean rank). Best and second-best ranks, determined before rounding, are bold and underlined, respectively.

Method	ARI ↑	NMI ↑	VI ↓	Binder ↓
CoClusterPFN (metric-aligned)	**0.390 (3.22)**	**0.398 (3.48)**	1.102 (3.44)	**0.267 (3.67)**
TabClustPFN	0.344 (4.38)	0.351 (4.59)	1.264 (5.04)	0.283 (4.79)
ZEUS	0.330 (5.44)	0.349 (5.30)	1.278 (5.78)	0.293 (5.59)
TableDC	0.302 (5.38)	0.318 (5.11)	1.226 (5.06)	0.302 (5.41)
Ward	0.324 (4.58)	0.339 (5.07)	1.280 (5.58)	0.289 (4.83)
k-means++	0.334 (4.73)	0.338 (4.82)	1.303 (5.95)	0.287 (4.63)
GMM + BIC	0.318 (5.08)	0.341 (4.70)	1.236 (5.16)	0.295 (5.15)
Spectral + silhouette	0.322 (5.13)	0.341 (5.17)	1.258 (5.57)	0.288 (4.88)
Agglomerative + silhouette	0.178 (7.07)	0.206 (6.77)	**0.988 (3.42)**	0.404 (6.06)

4.3 Benchmark on CLUBench

We evaluate CoClusterPFN on CLUBench, a benchmark of tabular clustering datasets (Xiao et al., 2026). Because CoClusterPFN's training prior contains at most 10 clusters and 20 features, we use the 60 datasets satisfying these constraints. We cap datasets at 2,000 observations by uniformly subsampling larger datasets to more closely match our training regime. Our classical baselines include k-means++, Ward clustering, agglomerative clustering with silhouette-based linkage selection, Gaussian mixtures with AIC or BIC selection, and spectral clustering with silhouette-based neighborhood selection. See Appendix C for more details.

We also compare with three learned tabular clustering methods. TabClustPFN (Zhao et al., 2026b) is evaluated using its frozen pretrained model. For ZEUS (Marszałek et al., 2025), we apply k-means to representations from its frozen encoder. TableDC (Rauf et al., 2025) is optimized separately on each dataset. In the unknown-K setting, ZEUS selects the partition by embedding-space silhouette over $K \in \{2, \ldots, 10\}$. TableDC requires a supplied count and therefore only appears in one setting. We use the prespecified metric-specific decision rules described in Section 3 for CoClusterPFN; CLUBench was not used for their selection.

Table 2 reports the supplied-K results. Using the prespecified metric-aligned decision rules, CoClusterPFN obtains the best mean score and mean rank for ARI, NMI, and Binder loss, and the second-best mean rank for VI. For supplied K, CoClusterPFN's paired ARI improvement over each of the eight baselines has a 95% bootstrap interval excluding zero, and all eight two-sided Wilcoxon tests remain significant after Holm correction (adjusted $p \leq 0.0119$ for significance level $\alpha = 0.05$).

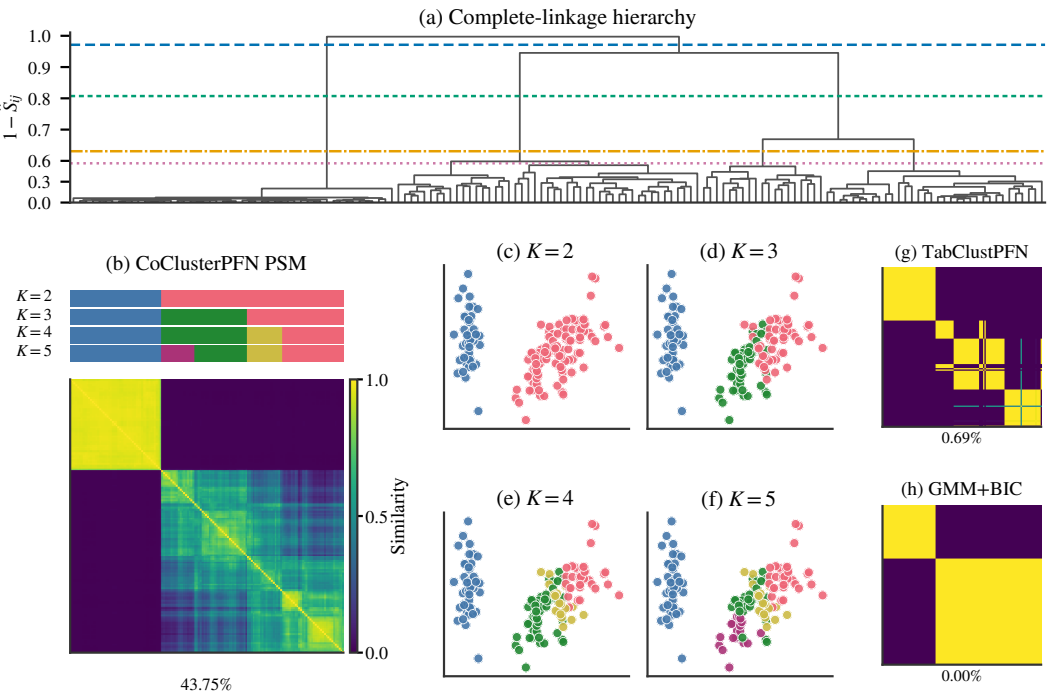


Figure 3: Iris case study of CoClusterPFN (with unknown K). Panel (a) shows the complete-linkage hierarchy of the CoClusterPFN PSM; the display expands dissimilarities above 0.6. Panel (b) shows the marginalized PSM with aligned strips for cuts at $K \in \{2, 3, 4, 5\}$; MaxPEAR selects $\widehat{K} = 3$. Panels (c)–(f) show these cuts in PCA space. Panels (g) and (h) show PSMs constructed from the responsibilities of TabClustPFN and GMM+BIC, which select $\widehat{K} = 4$ and $\widehat{K} = 2$, respectively. Percentages indicate the fraction of unordered off-diagonal pairs with similarities in $(0.1, 0.9)$. All similarity matrices use the same CoClusterPFN leaf order and shared $[0, 1]$ scale.

The favorable mean-score comparison does not depend entirely on using different decision rules across metrics: with supplied K, a single MaxPEAR partition per dataset improves all four mean scores relative to TabClustPFN (Table 12 in Appendix C.5).

The unknown-K results are qualitatively similar and given in Table 11 (Appendix C.4) and Table 7. CoClusterPFN obtains mean ARI 0.326, NMI 0.362, VI 1.168, and Binder loss 0.312. It again ranks first for ARI, NMI, and Binder loss and second for VI, while its ARI improvement remains significant after Holm correction against four of the seven baselines.

Predicting the correct number of clusters remains difficult for CoClusterPFN (Table 13, Appendix C.5). The ARI-selected MaxPEAR partitions recover the reference count on 12 of 60 datasets, overestimate it on 41, and underestimate it on 7, with a K MAE of 3.08. For estimating K, other approaches work better, and agglomerative clustering with silhouette-based selection has the lowest count MAE of 1.6.

Count diagnostics and full paired comparisons appear in Appendices C.5 and C.7, respectively. Appendix E ablates the effects of withholding K from the predicted PSM and from the decoder.

4.4 Interpreting uncertainty through a predicted PSM

One advantage of CoClusterPFN is the ability to inspect the PSM and analyze the different clusterings derived from it. The PSM also shows the uncertainty about whether each pair of observations belongs together. We illustrate this on Iris (Fisher, 1936) in a small case study. Principal component analysis (PCA) is used only for visualization. We run CoClusterPFN in the unknown-K prediction mode; see Figure 3 for the main case study results.

The PSM reveals one clearly separated 50-observation block: within-block co-clustering probabilities are high, while probabilities with the remaining observations are close to zero. The other 100 observations form a visibly less certain region with varying internal probabilities. This suggests that the less certain region supports further subdivisions, while the 50-observation block remains stable. MaxPEAR selects the $K = 3$ cut, producing relatively balanced clusters with an ARI of 0.87. Cuts at $K = 4$ and $K = 5$ further divide the groups within the less certain region. These are local refinements: the stable group persists, while only uncertain relationships are revised.

Panels (b), (g), and (h) of Figure 3 contrast the CoClusterPFN output with the PSMs derived from the responsibilities of TabClustPFN and GMM+BIC using Equation 4, with the cluster count withheld. Among unordered off-diagonal pairs, 44% of CoClusterPFN similarities are between 0.1 and 0.9, compared with only 0.7% for TabClustPFN and 0.0% for GMM+BIC. Their responsibilities are nearly one-hot, so the resulting PSMs largely reproduce the hard partition selected by each method and reveal little pairwise uncertainty, as expected. For more details, see Appendix D.

5 Limitations and Future Work

CoClusterPFN enforces symmetry and a unit diagonal but does not guarantee that its predictions are jointly coherent posterior marginals. Any valid PSM must satisfy $S_{ij} + S_{jk} - 1 \leq S_{ik}$ for every triple. At least one inequality was violated in 0.5–1.3% of tested triples for experiments on the tabular training prior, compared with 2.4–18.2% on CLUBench; complete results appear in Appendix C.3. Consistency constraints and broader partition searches may therefore improve clustering quality at additional computational cost (Wade & Ghahramani, 2018).

In the unknown-K setting, CoClusterPFN often does not correctly predict the number of clusters. This is consistent with Wade & Ghahramani (2018), who show that minimizing Binder's loss can overestimate the number of clusters; we suspect these PSM losses are not designed for cluster-count estimation. Using asymmetric losses may alleviate these problems (Dahl et al., 2022).

CoClusterPFN's all-object attention, pairwise head, and explicit PSM storage scale quadratically with N. Other amortized methods such as TabClustPFN offer a more scalable alternative when $K \ll N$ through $O(NK)$ cross-attention. To support further scaling, hierarchical clustering can be implemented in $O(N^2)$ time, and single linkage can compute similarities on demand with only $O(N)$ additional clustering workspace (Müllner, 2011). Faster attention and decoding may further reduce practical costs; however, it remains unclear how to move beyond $O(N^2)$ originating from the PSM.

A PSM summarizes pairwise uncertainty but does not determine the full posterior over partitions. A PFN that returns posterior samples may be a promising direction, supporting, for example, Monte Carlo estimation of posterior expected VI. It remains unclear how to design a fast PFN that can obtain posterior samples; autoregressive modeling or diffusion may offer promising directions.

Training the primary model required approximately 22 hours on two NVIDIA L40 GPUs. This cost is amortized across applications, but predictions may degrade under prior-data mismatch, and changing the prior requires retraining. MCMC may therefore be preferable when frequent prior changes are required. Distribution Transformers condition on explicit prior representations without retraining (Whittle et al., 2026); adapting this approach to clustering is a promising direction.

6 Conclusion

We introduced CoClusterPFN, a PFN that amortizes posterior similarity estimation. On synthetic 2D Gaussian mixtures, CoClusterPFN predicts posterior similarity matrices (PSMs) that closely agree with MCMC estimates and performs best on most reported criteria among the compared amortized clustering methods. Directly predicting co-clustering probabilities also improves decoded clustering relative to constructing a PSM from predicted responsibilities. Across 60 CLUBench datasets, CoClusterPFN achieves the best average ranks for ARI, NMI, and Binder loss, and the second-best for VI, both when the cluster count is supplied and when it is unknown. Finally, the Iris case study illustrates how the PSM reveals uncertain relationships and supports exploration of alternative clusterings through a dendrogram. Together, these results establish CoClusterPFN as an uncertainty-aware method that supports multiple clustering decisions with a single forward pass.

AI USE STATEMENT

We used generative AI tools to discuss conceptual framing and mathematical claims; design and critique experiments; assist with code and visualizations; review related literature; and draft and edit parts of the manuscript. Reported numerical results were produced by the experiments described in this paper, not generated as answers by an AI model. The authors reviewed AI-assisted contributions, verified the reported claims and results against sources and experimental outputs, and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The supplementary material provides an inference package containing an inference-only export of the primary CoClusterPFN checkpoint, known- and unknown-K prediction, the three PSM decoders used in the paper, and a small Iris example. The appendices document the prior, architecture, optimization and checkpoint selection, dataset manifest, and evaluation protocols, with hashes for the principal artifacts. The complete training and evaluation code will be released upon acceptance.

REFERENCES

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

Prajit Bhaskaran and Tom Viering. Transformers can do Bayesian clustering, 2025. arXiv:2510.24318.

David A. Binder. Bayesian cluster analysis. *Biometrika*, 65(1):31–38, 1978. doi: 10.1093/biomet/65.1.31.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Paul Edmund Chang, Nasrulloh Ratu Bagus Satrio Loka, Daolang Huang, Ulpu Remes, Samuel Kaski, and Luigi Acerbi. Amortized probabilistic conditioning for optimization, simulation and inference. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 703–711, 2025.

Samuel Coward, Erik Visse-Martindale, and Chithrupa Ramesh. Attention-based clustering: Learning a kernel from context. *arXiv preprint arXiv:2010.01040*, 2020.

David B. Dahl, Devin J. Johnson, and Peter Müller. Search algorithms and loss functions for Bayesian clustering. *Journal of Computational and Graphical Statistics*, 31(4):1189–1201, 2022. doi: 10.1080/10618600.2022.2069779.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002. doi: 10.1198/016214502760047131.

Arno Fritsch and Katja Ickstadt. Improved criteria for clustering based on the posterior similarity matrix. *Bayesian Analysis*, 4(2):367–392, 2009. doi: 10.1214/09-BA414.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

John W. Lau and Peter J. Green. Bayesian model-based clustering procedures. *Journal of Computational and Graphical Statistics*, 16(3):526–558, 2007. doi: 10.1198/106186007X238855.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019a.

Juho Lee, Yoonho Lee, and Yee Whye Teh. Deep amortized clustering. *arXiv preprint arXiv:1909.13433*, 2019b. doi: 10.48550/arXiv.1909.13433.

P. Li, F. Li, J. Wang, and Y. Qian. Clustering method for tabular data based on pretrained foundation models with synthetic data. *Journal of Computer Research and Development*, 62(9):2139–2151, 2025.

Huafeng Liu and Liping Jing. Amortized mixing coupling processes for clustering. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0851.

Patryk Marszałek, Tomasz Kuśmierczyk, Witold Wydmański, Jacek Tabor, and Marek Śmieja. ZEUS: Zero-shot embeddings for unsupervised separation of tabular data. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jeffrey W. Miller and Matthew T. Harrison. Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340–356, 2018. doi: 10.1080/01621459.2016.1255636.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

Ari Pakman, Yueqi Wang, Catalin Mitelut, Jinhyung Lee, and Liam Paninski. Neural clustering processes. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7455–7465. PMLR, 2020.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847, 2025.

Herilalaina Rakotoarison, Steven Adriaensen, Tom Viering, Carl Hvarfner, Samuel Müller, Frank Hutter, and Eytan Bakshy. α-PFN: Fast entropy search via in-context learning. In *Forty-third International Conference on Machine Learning*, 2026.

Hafiz Tayyab Rauf, André Freitas, and Norman W. Paton. TableDC: Deep clustering for tabular data. *Proceedings of the ACM on Management of Data*, 3(3):229:1–229:28, 2025. doi: 10.1145/3725366.

Arik Reuter, Tim G. J. Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full Bayesian inference in context? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51531–51582. PMLR, 2025. doi: 10.48550/arXiv.2501.16825.

Tommaso Rigon, Amy H. Herring, and David B. Dunson. A generalized Bayes framework for probabilistic clustering. *Biometrika*, 110(3):559–578, 2023. doi: 10.1093/biomet/asad004.

Meitar Ronen, Shahaf E. Finder, and Oren Freifeld. DeepDPM: Deep clustering with an unknown number of clusters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 62(4):795–809, 2000. doi: 10.1111/1467-9868.00265.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Sara Wade. Bayesian cluster analysis. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2247):20220149, 2023. doi: 10.1098/rsta.2022.0149.

Sara Wade and Zoubin Ghahramani. Bayesian cluster analysis: Point estimation and credible balls (with discussion). *Bayesian Analysis*, 13(2):559–626, 2018. doi: 10.1214/17-BA1073.

George Whittle, Juliusz Ziomek, Jacob Henry Rawling, and Michael A. Osborne. Distribution transformers: Fast approximate Bayesian inference with on-the-fly prior adaptation. In *Forty-third International Conference on Machine Learning*, 2026.

Feng Xiao, Dazhi Fu, Chris Ding, and Jicong Fan. CLUBench: A clustering benchmark. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 10044–10055, 2026.

Mingjie Zhao, Yunfan Zhang, Yiqun Zhang, and Yiu-ming Cheung. Beyond statistical co-occurrence: Unlocking intrinsic semantics for tabular data clustering. *arXiv preprint arXiv:2604.10865*, 2026a.

Tianqi Zhao, Guanyang Wang, Yan Shuo Tan, and Qiong Zhang. TabClustPFN: A prior-data fitted network for tabular data clustering. *arXiv preprint arXiv:2601.21656*, 2026b. doi: 10.48550/arXiv.2601.21656.

A TABULAR PRIOR DETAILS

This appendix provides additional details of the tabular class-conditional prior summarized in Section 3.

Cluster semantics. Under this prior, a latent class need not form one compact region in Euclidean feature space. It may be multimodal, observed features may be irrelevant or redundant, and class information may be expressed through continuous, categorical, ordinal, or count-valued variables. Before target-label perturbation, observations belong to the same cluster when they share the latent class that generated them.

For each task, we sample D and K_0. We reserve two observations for each class to guarantee minimum occupancy before the later label-reassignment step, sample class weights from a symmetric Dirichlet distribution, and allocate the remaining observations multinomially. This produces initial Dirichlet–multinomial class counts with a minimum occupancy of two observations. Conditional on these assignments, observations are generated using either the general class-conditional branch or the clean finite-Gaussian branch.

Latent class structure. In the general branch, observations are first generated in a latent space of dimension $d_{\mathrm{lat}} \leq D$. Each class is assigned a latent center, and the separation between class centers varies across tasks. Some tasks have globally reduced separation, while others contain deliberately close class pairs, allowing the class-conditional distributions to overlap substantially.

A class normally has one latent mode but may instead contain two or three subcomponents centered around its class center. All subcomponents share the same class label. Each has a randomly sampled full covariance matrix, allowing its scale and anisotropy to vary. Latent observations are Gaussian by default, while a Student-t scale mixture occasionally produces heavy tails. The resulting class-conditional distribution can therefore be elliptical, heavy-tailed, or multimodal before being mapped into the observed feature space.

Observed features. The latent vectors are mapped to informative continuous features through a task-specific random linear transformation with sparse coefficients. Observation noise is added after this transformation. Each informative feature is then either left unchanged or transformed using a hyperbolic tangent, signed logarithm, signed square, or sinusoid. These operations create nonlinear feature distributions and dependencies while generally preserving varying amounts of class information.

A sampled fraction of the observed-feature slots is reserved for auxiliary columns. Most are nuisance features generated independently of the class assignments using Gaussian, uniform, Laplace, categorical, or capped-Zipf distributions. With probability 0.15, an auxiliary column is instead a noisy duplicate of an informative column and is therefore redundant rather than irrelevant. Informative, nuisance, and duplicate columns are randomly permuted, so their roles cannot be identified from their positions.

Discrete features are created by transforming or replacing existing columns, so the number of observed features remains D. Independently for each eligible column, with probability 0.20 the column is replaced by a directly sampled discrete variable. Its type is binary, nominal, ordinal, or count-valued, selected with probabilities 0.36, 0.34, 0.20, and 0.10, respectively. Binary variables follow Bernoulli distributions with class-specific success probabilities; nominal variables follow class-specific categorical distributions with unordered category codes; ordinal variables are obtained by thresholding a Gaussian variable with a class-specific location; and counts follow either a Poisson distribution or an overdispersed gamma–Poisson mixture with a class-specific rate. The strength of the class dependence ranges from noise-like to nearly deterministic. Among columns not replaced in this way, with conditional probability 0.15 an existing column is quantized at empirical quantiles. In addition, 20% of tasks receive a class-dependent additive offset in one eligible column; columns sampled directly as discrete variables are excluded.

In the general branch, after feature generation, columns are standardized, a small fraction of entries is replaced by a fixed missing-value code, and each target assignment is selected for random reassignment with a small probability. The reassignment draw may select the observation's original class. These reassignments can empty an initial class; empty labels are removed, and $K \leq K_0$ is the final

number of occupied target classes. These mechanisms expose the model to irrelevant dimensions, redundancy, mixed feature types, missingness, and weakly inconsistent class signals, including data that depart from a clean continuous feature matrix.

Clean Gaussian branch. The remaining tasks are finite Gaussian mixtures generated directly in the observed feature space, with one Gaussian component per class. A task uniformly selects one covariance family—spherical, diagonal, or full—and each class receives separately sampled covariance parameters within that family. Component scales, covariance condition numbers, and the minimum separation between component centers vary across tasks. This branch ensures that the training prior also covers classical continuous clustering problems without the additional nuisance mechanisms of the general branch. After generation, its columns are standardized; the missingness and label-perturbation mechanisms of the general branch are not applied, so the target assignments coincide with the generating classes and $K = K_0$.

Numerical parameterization and design choices. Table 3 summarizes the selected tabular training configuration. Its numerical settings were specified manually as prior-design choices and were not selected through a systematic hyperparameter search. The probabilities were chosen so that each intended source of variation occurred often enough to affect training without dominating the generated tasks. During development, we inspected sampled datasets and synthetic prior-predictive diagnostics to check that the prior covered varied structures and difficulty levels without frequently producing degenerate tasks. We did not select the settings by optimizing aggregate CLUBench performance; model checkpoints and decoding rules were selected exclusively on synthetic validation tasks.

In the general branch, the auxiliary-feature target fraction f is sampled from the range reported in the table, and the number of auxiliary columns is $\min\{\mathrm{round}(fD), D-1\}$. The realized fraction can therefore differ from f, particularly for small D.

Table 3: Main tabular prior settings. Settings under the general-branch heading are conditional on selecting that branch, which occurs with probability 0.75. Feature-level probabilities are applied independently to eligible observed columns.

Quantity	Setting
Observation-count band	Uniform over four bands
Observations within selected band	Discrete uniform 64–256 / 257–512 513–1024 / 1025–2048
Ambient dimension	$D \sim \mathrm{Uniform}\{2, \ldots, 20\}$
Initial number of classes	$K_0 \sim \mathrm{Uniform}\{2, \ldots, 10\}$
Initial minimum class size	2 observations (before reassignment)
Symmetric Dirichlet concentration	3.0
Preprocessing	column standardization
Per-task clean-Gaussian branch probability	0.25
General class-conditional branch (conditional settings)	
Latent dimension	$2, \ldots, \min(6, D)$
Per-class subcomponent probability	0.50; at most 3 modes
Per-subcomponent Student-t probability	0.35
Per-task close-pair probability	0.45
Per-task global center-scale reduction probability	0.50
Sampled auxiliary-feature target fraction	0.20–0.40
Per-informative-feature transformation-draw probability	0.35
Per-informative-feature non-identity probability	0.28
Per-observed-column direct-discrete replacement	0.20
Conditional quantization among non-replaced columns	0.15
Direct-discrete type probabilities	0.36 / 0.34 / 0.20 / 0.10
Per-auxiliary-column duplicate probability	0.15
Per-task class-offset probability	0.20
Per-entry missingness probability	0.01
Per-observation reassignment draw probability	0.01

B Additional method details

B.1 Architecture implementation

Unlabeled TabICL row encoder. We initialize the input stage from the pretrained TabICL model (Qu et al., 2025). We use its column-embedding and row-interaction modules but remove the supervised in-context prediction stage. No target labels are passed to TabICL. Instead, all N observations form one unlabeled table, and the encoder uses the complete set of rows to construct a fixed-width representation

$$e_i = E_\psi(X)_i \in \mathbb{R}^{d_E} \tag{5}$$

for each observation. This stage allows a row's representation to depend on the empirical distributions and dependencies of the input features rather than treating every scalar entry independently. Because TabICL applies shared feature-processing modules across columns and aggregates a variable number of columns into a fixed-width row embedding, CoClusterPFN is not tied to a fixed input dimensionality. The same architecture and parameterization can process tables with different numbers of observed features without requiring a dimension-specific input projection.

The TabICL modules remain trainable. We optimize them jointly with the rest of CoClusterPFN, but use a learning rate 100 times smaller than that of the newly initialized layers. The smaller rate permits adaptation from supervised tabular pretraining to unlabeled clustering while limiting destructive changes to the pretrained representation.

Set-contextualized point representations. A learned linear map $W \in \mathbb{R}^{d_H \times d_E}$ projects each d_E-dimensional TabICL row embedding to the model width d_H. A Transformer encoder then processes

all projected rows jointly (Vaswani et al., 2017):

$$(h_1, \ldots, h_N) = T_\theta(We_1, \ldots, We_N), \qquad h_i \in \mathbb{R}^{d_H}. \tag{6}$$

Through self-attention, h_i can depend on every observation in the dataset. This allows the model to use dataset-level context, including cases in which the relevance of a feature is apparent only from its distribution across rows.

Neither the TabICL modules nor the added Transformer uses row-position encodings. The complete encoder is therefore equivariant to permutations of the input observations: permuting the rows of X applies the same permutation to the states $(h_1, \ldots, h_N)$ (Lee et al., 2019a). Because the subsequent head is shared across pairs, the predicted PSM is permuted consistently in both matrix dimensions.

Known and unspecified cluster counts. The code $q(C)$ has $K_{\max}+1$ entries. When K is available, $q(C)$ is its one-hot encoding for $K \in \{1, \ldots, K_{\max}\}$. When the count is unspecified, the zeroth entry is active and the remaining entries are zero. These are the *known-K* and *unknown-K* prediction modes, respectively.

The count code is appended only to the pair features after the point states h_i have been computed. The TabICL encoder and set Transformer therefore produce the same representation of X in both modes. Supplying K changes how the pairwise head interprets a pair of contextualized states; it does not change the states themselves. This lets one encoding support both conditional and marginalized predictions and isolates the effect of supplying K to the final probability mapping. The late conditioning is not an explicit summation over candidate values of K. Both modes are trained on the same sampled pairs and targets; Appendix B.4 gives the training objectives and corresponding population targets.

Implementation dimensions and PSM construction. In the primary tabular model, TabICL returns $d_E = 512$ features per row and the projection maps them to $d_H = 256$. The set encoder contains four Transformer layers with four attention heads and a feed-forward width of 512. The pairwise head contains two hidden layers of width 256 with GELU activations.

During training, the pairwise head is evaluated only for stratified samples of off-diagonal pairs, as described in Appendix B.4. During inference, the dataset is encoded once, after which the pairwise head evaluates all N^2 ordered pairs, including the diagonal, in chunks. Let $\tilde{S}$ denote the raw pairwise-head output. The resulting matrix is averaged with its transpose, and its diagonal is then set to one:

$$\widehat{S}_{ij} = \begin{cases} 1, & i = j, \\ \frac{1}{2}(\widetilde{S}_{ij} + \widetilde{S}_{ji}), & i \neq j. \end{cases} \tag{7}$$

Thus, although the pair features are analytically symmetric, the inference adapter explicitly enforces the symmetry and diagonal constraints.

Materializing the complete PSM necessarily requires $O(N^2)$ outputs and memory, and the pairwise head performs $O(N^2)$ evaluations. Chunking bounds the intermediate memory used by the pairwise head but does not change this total output or computation scaling. By comparison, a responsibility-based model that returns probabilities for K cluster slots materializes $O(NK)$ probabilities, although the complexity of its shared encoder may differ.

B.2 Primary optimization and checkpoint selection

Optimization. The primary CoClusterPFN model was trained for 89,600 optimizer steps using AdamW. At each step, one of four observation-count bands was selected with equal probability: 64–256, 257–512, 513–1024, or 1025–2048. The observation count N was then sampled uniformly from the integers in the selected band and shared by all tasks in that update. The corresponding global batch sizes were 128, 64, 32, and 16 tasks, respectively. The expected batch size was therefore 60 tasks per update, corresponding to 5,376,000 generated training tasks in expectation over the complete schedule.

Training was distributed over two GPUs, with each global batch divided equally between the two ranks. We used learning rates of 10^{-4} for the newly initialized CoClusterPFN parameters and 10^{-6} for the pretrained TabICL encoder. The learning rates were linearly warmed up for 4,480 steps and

then cosine-decayed to zero. Gradients were clipped to a global norm of 1.0. AdamW otherwise used the PyTorch defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.01. Training used random seed 1.

Checkpoint selection. Checkpoints were evaluated every 500 training steps. Validation used a fixed collection of 256 synthetic tasks generated with seed 20260923, with 64 tasks from each of the four training observation-count bands. For the mixed-K model, the selection criterion was the equally weighted mean of the known-K and unknown-K natural pairwise BCE losses.

The minimum validation criterion was 0.295447 at logged step 87,500. Because the log labels are zero-based while checkpoints store the number of completed steps, this validation is stored in the checkpoint with epoch counter 87,501. CLUBench data and labels were not used for checkpoint selection.

Frozen checkpoint and training environment. The primary results use the checkpoint attaining the minimum validation criterion reported above. Training used PyTorch 2.2.2 with CUDA 12.1 in an Apptainer environment on two NVIDIA L40 GPUs and took 21 hours and 56 minutes.

Primary training checkpoint SHA-256:
`4fa6307cebe18497802cafc58daf3c4892d711b714d2946f1a458378aec9e89f`
Released inference checkpoint SHA-256:
`13f9d3b3a2497dfe4d61ba2212a0add8e3e858927ef6fe31d264ec24800af4ff`
Pretrained TabICL checkpoint SHA-256:
`bdc7dbd5e4ff21f8f0456fcf90c6b7cdf72dbea960f2d05b19bec19f9b3d4ed0`

The released checkpoint is an inference-only export of the primary training checkpoint. Its retained tensors are bitwise identical; training state, run metadata, and unused count and responsibility heads are omitted.

B.3 Responsibility-ablation objective

The responsibility model predicts K active logits ℓ_{ic} for each observation. For each task independently, we construct the cost matrix

$$C_{ac} = -\sum_{i:z_i=a} \log([\mathrm{softmax}(\ell_i)]_c)\,, \tag{8}$$

where a indexes the occupied target clusters and c indexes the active output slots. The Hungarian algorithm finds the one-to-one assignment

$$\pi^\star \in \arg\min_{\pi} \sum_{a} C_{a,\pi(a)}. \tag{9}$$

The training loss is $N^{-1}\sum_a C_{a,\pi^\star(a)}$, averaged across tasks. Matching is recomputed for every task and update and is restricted to the K active responsibility slots. At evaluation time, writing $r_i = \mathrm{softmax}(\ell_i)$, the implied PSM is

$$S_{ij}^{\mathrm{resp}} = \sum_{c=1}^{K} r_{ic} r_{jc}, \qquad i \neq j. \tag{10}$$

The training and checkpoint-selection protocol for this comparison is given in Appendix C.2.

B.4 Pair-sampling estimator

All-pairs objective and population target. For each synthetic task, we construct Y from the sampled assignments and consider the $\binom{N}{2}$ unordered off-diagonal pairs $i < j$. For a conditioning mode $m \in \{\mathrm{known}, \mathrm{unknown}\}$, the full all-pairs binary-cross-entropy objective is

$$\mathcal{L}_{\mathrm{all}}^{(m)} = \mathbb{E}_{(X,z)}\left[-\frac{1}{\binom{N}{2}} \sum_{i<j} \left(Y_{ij} \log \widetilde{S}_{ij}^{(m)} + (1 - Y_{ij}) \log(1 - \widetilde{S}_{ij}^{(m)}) \right) \right]. \tag{11}$$

Let $A_m = (X, K)$ for the known-K mode and $A_m = X$ for the unknown-K mode. For a fixed pair, write $p_m = P(Y_{ij} = 1 \mid A_m)$ and suppose the network predicts s. Its conditional expected loss is

$$\mathcal{R}(s \mid A_m) = -p_m \log s - (1 - p_m) \log(1 - s). \tag{12}$$

This function is uniquely minimized at $s = p_m$. Therefore, the population targets for the two conditioning modes are

$$\widetilde{S}_{ij}^{(\text{known}),\star} = P(z_i = z_j \mid X, K), \qquad \widetilde{S}_{ij}^{(\text{unknown}),\star} = P(z_i = z_j \mid X). \tag{13}$$

Stratified estimator. Let $n_1, \ldots, n_K$ be the target class sizes for one task. Its exact positive-pair rate is

$$\pi = \frac{\sum_{k=1}^{K} \binom{n_k}{2}}{\binom{N}{2}}. \tag{14}$$

For each task and training update, we sample with replacement $M_+ = 4096$ positive pairs with $z_i = z_j$ and $M_- = 4096$ negative pairs with $z_i \neq z_j$. Writing these samples as $\mathcal{S}_+$ and $\mathcal{S}_-$, respectively, we compute

$$\begin{aligned}
\widehat{\mathcal{L}}_+^{(m)} &= -\frac{1}{M_+} \sum_{(i,j) \in \mathcal{S}_+} \log \widetilde{S}_{ij}^{(m)}, \\
\widehat{\mathcal{L}}_-^{(m)} &= -\frac{1}{M_-} \sum_{(i,j) \in \mathcal{S}_-} \log\left(1 - \widetilde{S}_{ij}^{(m)}\right), \\
\widehat{\mathcal{L}}_{\text{pair}}^{(m)} &= \pi \widehat{\mathcal{L}}_+^{(m)} + (1 - \pi) \widehat{\mathcal{L}}_-^{(m)}.
\end{aligned} \tag{15}$$

Uniform sampling within each stratum makes the first two terms unbiased estimates of the positive- and negative-pair losses. Weighting them by the natural pair rate π therefore recovers the all-pairs objective in Equation (11) in expectation. Balanced sampling ensures substantial coverage of positive pairs without imposing an artificial 50% positive-pair rate, which would distort probability calibration.

Under the variable-size training schedule, a task contains between 64 and 2,048 observations and therefore between $\binom{64}{2} = 2{,}016$ and $\binom{2048}{2} = 2{,}096{,}128$ distinct unordered off-diagonal pairs. Regardless of N, we evaluate 4,096 sampled positive-pair instances and 4,096 sampled negative-pair instances per task. Sampling is performed with replacement, so duplicate pair instances are possible, particularly for the smaller datasets, and pairs are resampled at every update. The implementation samples endpoint orientations, but the symmetric pairwise head makes the orientation immaterial. Diagonal pairs are not sampled and are fixed to one when the complete PSM is constructed. For mixed training, the same sampled pairs and targets are used for both count codes, and their losses are combined with equal weight:

$$\widehat{\mathcal{L}}_{\text{mixed}} = \tfrac{1}{2} \widehat{\mathcal{L}}_{\text{pair}}^{(\text{known})} + \tfrac{1}{2} \widehat{\mathcal{L}}_{\text{pair}}^{(\text{unknown})}. \tag{16}$$

Degenerate-stratum handling. Initial class allocations contain at least two observations in each of $K_0 \geq 2$ classes. After label-noise reassignment, any clusters left without observations are removed, and the remaining cluster labels are renumbered consecutively. Pair sampling then draws 4,096 positive and 4,096 negative ordered endpoint pairs per task. Because the stratified objective requires both strata, a task containing only one occupied class is treated as invalid and raises an error rather than fabricating negative pairs or changing the loss. No such collapse occurred in the completed training or validation runs.

B.5 PSM DECODER OBJECTIVES

Metric-aligned decoder selection. Table 4 reports the rules selected on 256 held-out synthetic validation tasks generated independently of CLUBench. The validation collection contains exactly 64 tasks from each of the four observation-count bands used during primary training: 64–256, 257–512, 513–1024, and 1025–2048. Within each band, N is sampled uniformly from the included integers. For each reporting metric, the decoder is selected by maximizing mean ARI or NMI, or minimizing mean VI or N-invariant Binder loss, respectively.

Table 4: Metric-aligned PSM decoders selected on synthetic validation tasks.

Reporting metric	Known K	Unknown K
ARI	MaxPEAR	MaxPEAR
NMI	MinVI-LB	MaxPEAR
VI	MinVI-LB	MinVI-LB
N-invariant Binder loss	MinBinder	MinBinder

Candidate partitions. Let c be a candidate partition and define its co-membership indicators by

$$B_{ij}(c) = \mathbf{1}\{c_i = c_j\}\,. \tag{17}$$

We construct average- and complete-linkage hierarchies from the dissimilarities $1 - \widehat{S}_{ij}$. Let

$$\mathcal{H}(\widehat{S}) = \mathcal{H}_{\mathrm{avg}}(\widehat{S}) \cup \mathcal{H}_{\mathrm{complete}}(\widehat{S}) \tag{18}$$

denote the union of their candidate cuts. The admissible candidate sets for the fixed- and variable-K settings are

$$\begin{aligned} \mathcal{H}_K(\widehat{S}) &= \{c \in \mathcal{H}(\widehat{S}) : |c| = K\}, \\ \mathcal{H}_{2:10}(\widehat{S}) &= \{c \in \mathcal{H}(\widehat{S}) : 2 \le |c| \le 10\}, \end{aligned} \tag{19}$$

where $|c|$ is the number of groups in c.

MaxPEAR. MaxPEAR uses the PSM-based approximation to posterior expected adjusted Rand index proposed by Fritsch & Ickstadt (2009). Define

$$\begin{aligned} M &= \binom{N}{2}, \qquad A(c) = \sum_{i<j} B_{ij}(c), \\ U(\widehat{S}) &= \sum_{i<j} \widehat{S}_{ij}, \quad T(c;\widehat{S}) = \sum_{i<j} B_{ij}(c)\widehat{S}_{ij}. \end{aligned} \tag{20}$$

The MaxPEAR score is

$$Q_{\mathrm{MP}}(c;\widehat{S}) = \frac{T(c;\widehat{S}) - A(c)U(\widehat{S})/M}{\frac{1}{2}\{A(c) + U(\widehat{S})\} - A(c)U(\widehat{S})/M}, \tag{21}$$

with larger values preferred. The fixed-K and variable-K decisions are

$$\begin{aligned} \widehat{c}^{\,(\mathrm{known})}_{\mathrm{MP}} &\in \arg\max_{c \in \mathcal{H}_K(\widehat{S}^{(\mathrm{known})})} Q_{\mathrm{MP}}(c;\widehat{S}^{(\mathrm{known})}), \\ \widehat{c}^{\,(\mathrm{unknown})}_{\mathrm{MP}} &\in \arg\max_{c \in \mathcal{H}_{2:10}(\widehat{S}^{(\mathrm{unknown})})} Q_{\mathrm{MP}}(c;\widehat{S}^{(\mathrm{unknown})}). \end{aligned} \tag{22}$$

Variation-of-information lower bound. We also evaluate the PSM-based lower bound on posterior expected variation of information from Wade & Ghahramani (2018). Let $\mathcal{C}_i(c) = \{j : c_j = c_i\}$ denote the candidate cluster containing observation i. The implementation evaluates the equivalent reduced objective

$$Q^{\mathrm{red}}_{\mathrm{VI\text{-}LB}}(c;\widehat{S}) = \frac{1}{N}\sum_{i=1}^{N}\left[\ln|\mathcal{C}_i(c)| - 2\ln\left(\sum_{j\in\mathcal{C}_i(c)}\widehat{S}_{ij}\right)\right], \tag{23}$$

with smaller values preferred. We optimize the candidate-dependent part of the VI lower-bound criterion using natural logarithms, omitting terms independent of the candidate partition. Neither the logarithm base nor the omitted terms affect candidate rankings. We apply this score to the same fixed-K and variable-K candidate sets as MaxPEAR.

Binder loss. With equal cost for a false merge and a false split, the plug-in posterior expected Binder criterion is

$$R_{\mathrm{B}}(c;\widehat{S}) = \sum_{i<j}\Big[B_{ij}(c)(1-\widehat{S}_{ij}) + (1-B_{ij}(c))\widehat{S}_{ij}\Big]. \tag{24}$$

The first term charges the candidate partition when it joins a pair that the posterior tends to separate. The second term charges it when it splits a pair that the posterior tends to join. Multiplying this criterion by $2/N^2$ gives the N-invariant scale used for benchmark reporting in Appendix C; this candidate-independent scaling does not change the minimizing partition.

The ideal plug-in Binder action under the predicted PSM is any partition that globally minimizes this criterion:

$$c^{\star}_{\mathrm{B}} \in \arg\min_{c\in\mathcal{C}_N} R_{\mathrm{B}}(c;\widehat{S}), \tag{25}$$

where $\mathcal{C}_N$ is the set of valid partitions. The implemented MinBinder decoder instead restricts this search to the hierarchical candidates:

$$\begin{aligned} \widehat{c}^{\,(\mathrm{known})}_{\mathrm{B}} &\in \arg\min_{c\in\mathcal{H}_K(\widehat{S}^{(\mathrm{known})})} R_{\mathrm{B}}(c;\widehat{S}^{(\mathrm{known})}),\\ \widehat{c}^{\,(\mathrm{unknown})}_{\mathrm{B}} &\in \arg\min_{c\in\mathcal{H}_{2:10}(\widehat{S}^{(\mathrm{unknown})})} R_{\mathrm{B}}(c;\widehat{S}^{(\mathrm{unknown})}). \end{aligned} \tag{26}$$

This restricted search returns a valid partition but does not guarantee the global optimum in Equation (25).

C Tabular evaluation details

C.1 Primary 60-dataset protocol

The primary tabular evaluation includes every available CLUBench tabular dataset with between 2 and 10 reference classes and at most 20 input features, matching the cluster-count and dimensionality ranges of the training prior. Applying these criteria to the 103 available tabular datasets yields the 60 datasets evaluated in Sections 4.2 and 4.3. No clustering-tendency or reference-label-alignment score is used to select this primary collection.

The evaluated subset is specified completely by the accompanying schema-v1 manifest. For every dataset, the manifest records the filename, source-file hash, original and evaluated sample counts, dimensionality, reference K, sampling seed, exact selected row indices, and hashes of the selected indices and labels.

> Manifest: `benchmarks/clubench_paper_v1.json`
> SHA-256: `abe92eb55dac6c11f5311b8f6d639b322311f637acb6b24cd6765bafb1ad5615`

The frozen evaluation manifest caps each dataset at 2,000 observations. We include all observations for the 41 datasets at or below this limit. For each of the remaining 19 datasets, we uniformly sample 2,000 observations without replacement and without using the reference labels. For reproducibility, the dataset at position (i) in the original CLUBench tabular manifest is sampled using NumPy's default_rng with seed $(1009i)$. We store the resulting indices in the evaluation manifest, ensuring that every method is evaluated on exactly the same observations.

Table 5 lists the included datasets and their original and evaluated sample counts, numbers of features, and reference cluster counts.

Table 5: The 60 CLUBench datasets included in the evaluation. N_{source} is the original number of observations, N is the number evaluated after the label-independent cap at 2,000 observations, D is the number of input features, and K is the reference number of clusters.

Dataset	N_{source}	N	D	K
echocardiogram.npz	61	61	10	2
skillcraft1_master_table_dataset.npz	3,303	2,000	18	6
breast_cancer_wisconsin_original.npz	683	683	9	2
smoker_condition.npz	1,012	1,012	7	2
glass_identification.npz	214	214	9	6
statlog_image_segmentation.npz	2,310	2,000	19	7
planning_relax.npz	182	182	12	2
customer_classification.npz	1,000	1,000	11	4
pima_indians_diabetes_database.npz	768	768	8	2
mobile_price_classification.npz	2,000	2,000	20	4
rice_seed_gonen_jasmine.npz	9,999	2,000	10	2
heart_attack_analysis_prediction_dataset.npz	303	303	13	2
user_knowledge_modeling.npz	258	258	5	4
world12d.npz	150	150	12	5
pumpkin_seeds.npz	2,500	2,000	12	2
iris.npz	150	150	4	3
wine.npz	178	178	13	3
mammographic_mass.npz	830	830	5	2
breast_tissue.npz	106	106	9	6
hepatitis.npz	80	80	19	2
predicting_pulsar_star.npz	9,273	2,000	8	2
wireless_indoor_localization.npz	2,000	2,000	7	4
zoo.npz	101	101	16	7
htru2.npz	9,999	2,000	8	2
rice_dataset_cammeo_and_osmancik.npz	3,810	2,000	7	2
banknote_authentication.npz	1,372	1,372	4	2
wine_quality.npz	4,873	2,000	11	5
cardiovascular_study.npz	2,927	2,000	15	2
boston.npz	154	154	13	3
dry_bean.npz	9,997	2,000	16	7
credit_risk_classification.npz	976	976	11	2
website_phishing.npz	1,353	1,353	9	3
siberian_weather_stats.npz	1,407	1,407	11	7
orbit_classification_for_prediction_nasa.npz	1,722	1,722	11	3
magic_gamma_telescope.npz	9,999	2,000	10	2
raisin.npz	900	900	7	2
patient_treatment_classification.npz	4,412	2,000	10	2
paris_housing_classification.npz	10,000	2,000	17	2
seeds.npz	210	210	7	3
wine_customer.npz	178	178	13	3
classification_in_asteroseismology.npz	1,001	1,001	3	2
birds_bones_and_living_habits.npz	413	413	10	6
image_segmentation.npz	210	210	19	7
water_quality.npz	2,011	2,000	9	2
harbermans_survival.npz	306	306	3	2
yeast.npz	1,459	1,459	8	8
heart_disease.npz	297	297	13	5
ecoli.npz	327	327	7	5
breast_cancer_coimbra.npz	116	116	9	2
human_stress_detection.npz	2,001	2,000	3	3
pen_based_recognition_of_handwritten_digits.npz	7,494	2,000	16	10
diabetic_retinopathy_debrecen.npz	1,151	1,151	19	2
blood_transfusion_service_center.npz	748	748	4	2
mfeat-morphological.npz	2,000	2,000	6	10

Continued on next page

Table 5 continued

Dataset	N_{source}	N	D	K
JapaneseVowels.npz	9,961	2,000	14	9
rmftsa_sleepdata.npz	1,024	1,024	2	4
DrugConsumption.npz	1,749	1,749	12	4
shuttle.npz	10,000	2,000	9	2
vehicle.npz	846	846	18	4
poker-hand.npz	10,000	2,000	10	2

All preprocessing is fitted after applying these indices. For CoClusterPFN and the classical baselines, non-finite numeric entries are imputed with the arithmetic mean of their feature, computed using only the selected observations of that dataset. Entirely missing columns are set to zero and subsequently removed as constant; other columns with variance at most 10^{-12} are also removed. The remaining features are standardized using their dataset mean and population standard deviation. None of the 60 evaluated datasets contains non-finite values, so the reported results do not depend on the imputation rule. TabClustPFN uses its official preprocessing pipeline with z-normalization applied exactly once.

Reference labels are used only to determine dataset eligibility, supply K in known-K evaluations, and compute external evaluation metrics. In unknown-K evaluations, the methods and their selection criteria receive neither the reference labels nor their count. CoClusterPFN's checkpoint and decoders are selected using synthetic validation tasks.

Classical baseline specifications. Table 6 gives the complete fixed settings, search grids, selection criteria, and tie-breaking rules for the classical baselines. All stochastic baseline estimators use evaluation seed 0.

Table 6: Frozen classical baseline specifications. In the unknown-K setting, candidate counts are restricted by the number of observations as shown.

Baseline	Fixed settings and search grid	Selection
k-means	k-means++ initialization, Lloyd updates, 10 initializations, and 500 iterations; unknown $K \in \{2, \ldots, \min(10, N-1)\}$	Known K: supplied count. Unknown K: separate Euclidean-silhouette and Calinski–Harabasz pipelines; smaller K breaks ties.
Ward	Euclidean Ward linkage with the full hierarchy	Supplied K.
Agglomerative	Ward, average, complete, or single linkage; unknown $K \in \{2, \ldots, \min(10, N-1)\}$	Known K: Euclidean silhouette selects linkage at the supplied count. Unknown K: silhouette jointly selects linkage and count; ties are resolved by smaller K and then the listed linkage order.
GMM	Five initializations, 500 iterations, covariance regularization 10^{-6}, and full, tied, diagonal, or spherical covariance	Known K: BIC selects covariance. Unknown K: AIC or BIC jointly selects covariance and $K \in \{2, \ldots, \min(10, N)\}$.
Spectral	Nearest-neighbor affinity with $n_{\text{neighbors}} \in \{5, 10, 20, 50\}$ clipped to $N-1$; k-means assignment with 10 initializations	Euclidean silhouette at the supplied K.

Additional learned baselines. ZEUS uses the official frozen checkpoint and numerical preprocessing, producing a 512-dimensional zero-shot embedding. We apply k-means with 100 initializations, using the supplied count or selecting $K \in \{2, \ldots, 10\}$ by embedding-space silhouette. TableDC is initialized with BIRCH and trained separately on each standardized dataset for 30 autoencoder-pretraining epochs and 200 joint-clustering epochs. It receives the reference count but not the pointwise labels.

Evaluation software and frozen neural baseline. Evaluation used Python 3.12.10, NumPy 2.4.3, SciPy 1.17.1, scikit-learn 1.8.0, PyTorch `2.11.0+cu126`, pandas 3.0.1, and TabICL 2.1.1. The frozen TabClustPFN artifact identifiers are:

Git revision: `5673beaab72d69bb9db30da2740aef90975a3e0c`
Checkpoint SHA-256:
`feb7ab7ad21cd4702b57ce92338ace65f0d852ef1bc00653f2d7049a235095a2`

Evaluation metrics. For one dataset, let z denote the reference partition and c the predicted partition. Let $n_{ab} = |\{i : z_i = a, c_i = b\}|$ be their contingency counts, with $n_{a\cdot} = \sum_b n_{ab}$, $n_{\cdot b} = \sum_a n_{ab}$, and $N = \sum_{a,b} n_{ab}$. Writing $p_{ab} = n_{ab}/N$, $p_a = n_{a\cdot}/N$, and $q_b = n_{\cdot b}/N$, we use natural logarithms to define

$$H(z) = -\sum_a p_a \ln p_a, \qquad H(c) = -\sum_b q_b \ln q_b, \tag{27}$$

$$I(z;c) = \sum_{a,b:p_{ab}>0} p_{ab} \ln\left(\frac{p_{ab}}{p_a q_b}\right). \tag{28}$$

The reported normalized mutual information uses arithmetic normalization,

$$\mathrm{NMI}(z,c) = \frac{2I(z;c)}{H(z)+H(c)}, \tag{29}$$

and the reported variation of information is

$$\mathrm{VI}(z,c) = H(z) + H(c) - 2I(z;c), \tag{30}$$

measured in nats. This evaluation metric is the exact VI between two hard partitions; the implementation clips it to zero only to remove possible tiny negative values caused by floating-point error. It is distinct from the PSM-based VI lower-bound criterion used for decoding in Appendix B.5.

Let $D(z,c)$ be the number of unordered observation pairs on which the two partitions disagree about co-membership. We report the N-invariant Binder loss

$$\mathrm{Binder}_{N\text{-inv}}(z,c) = \frac{2}{N^2}\sum_{i<j} \mathbf{1}\{\mathbf{1}\{z_i = z_j\} \neq \mathbf{1}\{c_i = c_j\}\}. \tag{31}$$

Thus, $\mathrm{Binder}_{N\text{-inv}}(z,c) = 2D(z,c)/N^2$. It differs from pair-normalized Binder loss by the factor $(N-1)/N$: $2D/N^2 = [D/\binom{N}{2}](N-1)/N$. ARI is the standard adjusted Rand index. For each benchmark setting and metric, methods are ranked separately within each dataset in the favorable direction: descending for ARI and NMI and ascending for VI and N-invariant Binder loss. Average ranks are assigned to exact ties, and mean rank is the arithmetic mean of the per-dataset ranks over the 60 datasets. The complete metric-aligned tables report a separate mean rank for each metric. Finally, writing K_d for the number of clusters in the reference partition z_d and letting $\widehat{K}_d$ denote the cluster count selected or predicted by the method, exact-count accuracy is $60^{-1}\sum_d \mathbf{1}\left\{\widehat{K}_d = K_d\right\}$ and count MAE is $60^{-1}\sum_d |\widehat{K}_d - K_d|$.

Paired ARI comparisons. Table 7 reports paired comparisons between CoClusterPFN and every baseline. Differences are computed as CoClusterPFN ARI minus baseline ARI on each dataset. The intervals use 20,000 paired dataset-level bootstrap resamples, and wins/ties/losses count datasets on which CoClusterPFN has higher, equal, or lower ARI, respectively. We also use two-sided paired Wilcoxon signed-rank tests with Pratt handling of zero differences and apply Holm correction across baselines separately within each count setting. All eight supplied-K comparisons remain significant after correction (largest adjusted $p = 0.0119$), as do four of the seven unknown-K comparisons.

C.2 Matched-prior responsibility ablation

We additionally evaluated CoClusterPFN and the Responsibility-PFN on 300 independent tasks with 256 observations drawn from the exact tabular prior recorded in their checkpoints: 219 general-branch draws and 81 clean-Gaussian draws. The held-out tasks are shared across the two models, with no

Table 7: Paired ARI differences between CoClusterPFN and each frozen baseline. Positive differences favor CoClusterPFN.

Setting	Baseline	Mean Δ	95% interval	W/T/L
Known K	TabClustPFN	$+0.045237$	$[0.007926, 0.085880]$	39/0/21
Known K	k-means++	$+0.055859$	$[0.019040, 0.095744]$	41/0/19
Known K	Ward	$+0.065485$	$[0.026315, 0.109174]$	39/0/21
Known K	Spectral + silhouette	$+0.068037$	$[0.030975, 0.106423]$	42/0/18
Known K	ZEUS	$+0.059833$	$[0.020560, 0.101551]$	46/1/13
Known K	TableDC	$+0.087818$	$[0.050670, 0.127910]$	44/0/16
Known K	GMM + BIC covariance	$+0.071226$	$[0.024615, 0.123589]$	42/2/16
Known K	Agglomerative + silhouette	$+0.212019$	$[0.141655, 0.288710]$	52/0/8
Unknown K	k-means++ + Calinski–Harabasz	$+0.022076$	$[-0.022970, 0.068526]$	36/0/24
Unknown K	k-means++ + silhouette	$+0.039873$	$[-0.007302, 0.090618]$	39/0/21
Unknown K	TabClustPFN	$+0.045770$	$[0.019320, 0.074388]$	39/0/21
Unknown K	ZEUS + silhouette	$+0.035506$	$[0.000896, 0.071469]$	39/0/21
Unknown K	GMM + BIC	$+0.092167$	$[0.054235, 0.131398]$	42/0/18
Unknown K	GMM + AIC	$+0.114755$	$[0.073615, 0.158702]$	42/0/18
Unknown K	Agglomerative + silhouette	$+0.227434$	$[0.164316, 0.295722]$	57/0/3

additional input scaling. Probability scores use every unordered off-diagonal pair, and ARI uses the same known-K decoders as the CLUBench comparison. Paired intervals resample tasks.

The direct arm uses the primary CoClusterPFN checkpoint described in Appendix B.2. The Responsibility-PFN uses the same pretrained TabICL checkpoint, backbone, tabular prior, observation-count schedule, number of optimization steps, and training seed 1. Both models were trained for 89,600 optimizer steps.

Both checkpoints were selected using the fixed 256-task validation collection generated with seed 20260923. CoClusterPFN was selected at logged step 87,500 with validation criterion 0.295447. The Responsibility-PFN was selected at logged step 85,500 with permutation-matched validation cross-entropy 0.868397. These loss magnitudes are not directly comparable because the two models optimize different objectives.

Table 8: Direct PSM versus responsibility supervision on 300 fresh tasks from the matched tabular training prior. Both models receive the reference K. Differences are direct PSM minus responsibilities.

Metric	Direct PSM	Responsibilities	Difference	95% interval
PSM BCE ↓	**0.3051**	0.3337	-0.0286	$[-0.0334, -0.0242]$
PSM Brier ↓	**0.0966**	0.1065	-0.0099	$[-0.0117, -0.0083]$
PSM MAE ↓	**0.1925**	0.2100	-0.0175	$[-0.0205, -0.0147]$
Average-linkage ARI ↑	**0.5079**	0.4622	$+0.0457$	$[0.0371, 0.0550]$
MinBinder ARI ↑	**0.5056**	0.4669	$+0.0387$	$[0.0301, 0.0482]$
MaxPEAR ARI ↑	**0.5101**	0.4673	$+0.0428$	$[0.0343, 0.0521]$

Table 8 shows that direct PSM supervision improves all three pairwise prediction metrics and known-K decoded ARI under all three decision rules. All six paired 95% intervals exclude zero in the favorable direction. The targets in this experiment are sampled binary co-membership labels rather than exact posterior probabilities; the matched-GMM study in Section 4.1 provides the posterior-reference comparison.

C.3 PSM TRANSITIVITY DIAGNOSTIC

We quantify violations of the necessary PSM transitivity condition on the same 300 held-out tasks used in the responsibility ablation and on the 60 CLUBench datasets. We evaluate both the supplied- and unknown-K predictions. Table 9 reports the results. Violations are uncommon on tasks from the tabular training prior, but occur more frequently on CLUBench, particularly when K is supplied.

Table 9: Transitivity violations in CoClusterPFN PSMs. For each matrix, we evaluate up to 100,000 uniformly sampled unordered triples and all three inequalities associated with each triple; all triples are used when fewer are available. A triple is counted as violating when at least one of its three inequalities fails. Rates are dataset-level means with standard errors in parentheses, using a tolerance of 10^{-6}. Mean excess is pooled over all positive excesses.

Evaluation	K	Violating triples (%)	Mean excess
Held-out tabular prior	Supplied	1.29 (0.26)	0.029
Held-out tabular prior	Unspecified	0.51 (0.09)	0.031
CLUBench	Supplied	18.23 (2.89)	0.107
CLUBench	Unspecified	2.39 (0.66)	0.052

C.4 Complete metric-aligned leaderboards

The evaluation contains results for all nine supplied-K pipelines and all eight unknown-K pipelines on all 60 datasets, with no missing evaluations. Mean scores and mean ranks give equal weight to each dataset. Tables 10 and 11 report the complete results.

CoClusterPFN uses decision rules selected on 256 independent synthetic validation tasks. For supplied K, these are MaxPEAR for ARI, MinVI-LB for NMI and VI, and MinBinder for Binder loss. For unknown K, MaxPEAR is also used for NMI. Consequently, the CoClusterPFN entries within a leaderboard need not correspond to one common partition. All baseline pipelines remain fixed.

Table 10: Complete supplied-K CLUBench leaderboard. Entries are mean score (mean dataset-level rank); lower ranks are better. Best and second-best ranks, determined before rounding, are bold and underlined. Binder denotes N-invariant Binder loss.

Method	ARI ↑	NMI ↑	VI ↓	Binder ↓
CoClusterPFN (metric-aligned)	**0.389543 (3.217)**	**0.397542 (3.483)**	1.101752 (3.442)	**0.266545 (3.667)**
TabClustPFN	0.344306 (4.375)	0.350967 (4.592)	1.263723 (5.042)	0.283068 (4.792)
Ward	0.324058 (4.583)	0.338622 (5.067)	1.280436 (5.583)	0.289317 (4.825)
k-means++	0.333684 (4.733)	0.338389 (4.817)	1.302605 (5.950)	0.287437 (4.633)
GMM + BIC covariance	0.318317 (5.075)	0.341164 (4.700)	1.236184 (5.158)	0.295134 (5.150)
Spectral + silhouette neighbors	0.321506 (5.133)	0.341356 (5.167)	1.257707 (5.567)	0.288438 (4.875)
TableDC	0.301725 (5.375)	0.317587 (5.108)	1.226114 (5.058)	0.302362 (5.408)
ZEUS	0.329710 (5.442)	0.348982 (5.300)	1.278278 (5.783)	0.292955 (5.592)
Agglomerative + silhouette linkage	0.177524 (7.067)	0.206278 (6.767)	**0.988342 (3.417)**	0.403573 (6.058)

Table 11: Complete unknown-K CLUBench leaderboard. Entries are mean score (mean dataset-level rank); lower ranks are better. Best and second-best ranks, determined before rounding, are bold and underlined. Binder denotes N-invariant Binder loss.

Method	ARI ↑	NMI ↑	VI ↓	Binder ↓
CoClusterPFN (metric-aligned)	**0.325941 (3.100)**	**0.362138 (2.967)**	1.167639 (3.150)	**0.312337 (3.367)**
TabClustPFN	0.280171 (3.883)	0.312996 (3.900)	1.475435 (4.617)	0.344901 (4.083)
k-means++ + Calinski–Harabasz	0.303865 (3.983)	0.321123 (4.100)	1.270927 (3.992)	0.339245 (4.158)
k-means++ + silhouette	0.286069 (4.125)	0.303467 (4.508)	1.311017 (4.083)	0.346758 (4.367)
GMM + AIC	0.211186 (4.592)	0.307027 (4.358)	2.019910 (6.958)	0.362410 (4.842)
GMM + BIC	0.233775 (4.592)	0.315482 (4.425)	1.831287 (6.167)	0.362878 (4.867)
ZEUS + silhouette	0.290436 (4.692)	0.324139 (4.658)	1.306612 (4.092)	0.346262 (4.533)
Agglomerative + silhouette	0.098507 (7.033)	0.111968 (7.083)	**1.022614 (2.942)**	0.496738 (5.783)

C.5 Decoder sensitivity and count recovery

Decision-rule sensitivity. Table 12 reports each primary PSM decision rule separately. Each row therefore evaluates one partition per dataset across all four metrics. These results are descriptive: the metric-aligned rules used in the primary evaluation were selected beforehand on synthetic validation tasks and were not chosen using CLUBench labels.

Table 12: Sensitivity of CoClusterPFN to the PSM decision rule on the 60 compatible CLUBench datasets. Each row uses one decoded partition per dataset for all four metrics. Binder denotes N-invariant Binder loss.

Setting	Decoder	Selected for	ARI ↑	NMI ↑	VI ↓	Binder ↓
Known K	MaxPEAR	ARI	0.3895	0.3990	1.1459	0.2728
Known K	MinVI-LB	NMI, VI	0.3847	0.3975	1.1018	0.2707
Known K	MinBinder	Binder	0.3881	0.3973	1.1438	0.2665
Unknown K	MaxPEAR	ARI, NMI	0.3259	0.3621	1.4521	0.3161
Unknown K	MinVI-LB	VI	0.3470	0.3784	1.1676	0.3047
Unknown K	MinBinder	Binder	0.3184	0.3605	1.5094	0.3123

Using 20,000 paired dataset-level bootstrap resamples, the unadjusted 95% intervals for all pairwise ARI and NMI differences include zero. MinVI-LB yields lower VI than both alternatives in both count settings, with intervals excluding zero. For Binder loss, only the supplied-K MinBinder–MaxPEAR interval excludes zero.

Unknown-K count recovery. Table 13 compares count recovery for the ARI-selected unknown-K pipelines. CoClusterPFN uses MaxPEAR in the unknown-K setting; decision rules selected for other metrics can imply different counts.

Table 13: Unknown-K count recovery for the ARI-selected pipelines. Higher is better for exact-count accuracy, and lower is better for count MAE. Best and second-best values are bold and underlined.

Method	Exact K ↑	K MAE ↓
CoClusterPFN (MaxPEAR)	0.200	3.083
TabClustPFN	0.283	2.333
k-means++ + Calinski–Harabasz	<u>0.417</u>	1.683
k-means++ + silhouette	0.400	1.617
ZEUS + silhouette	0.383	<u>1.583</u>
GMM + BIC	0.100	4.150
GMM + AIC	0.050	5.783
Agglomerative + silhouette	**0.483**	**1.567**

C.6 RUNTIME PROTOCOL AND SCALING

Controlled neural inference comparison. We compare the primary CoClusterPFN checkpoint with the official TabClustPFN checkpoint and upstream implementation on an NVIDIA RTX 3090 using FP32 and PyTorch 2.11.0. The benchmark uses the exact (N, D, K) shapes of the 60 supplied-K CLUBench datasets. Each dataset receives one warm-up followed by five timed repetitions; CUDA is synchronized around each measurement. The timings exclude preprocessing, checkpoint loading, evaluation metrics, and CoClusterPFN's hard PSM decoder. They include input transfer to the device and transfer of the probability output back to the host. CoClusterPFN returns an $N \times N$ PSM, whereas TabClustPFN returns $N \times K$ responsibility probabilities.

Table 14 reports the size-stratified results. Over all datasets, the ratio of mean latencies is $184.2/35.9 = 5.1$, while the median of the 60 per-dataset ratios is 3.48. CoClusterPFN has lower measured latency on 18 datasets, all with $N \leq 306$. Across the five datasets with $61 \leq N \leq 116$, the mean measured latencies are 20.8 ms and 29.1 ms. Across the 22 datasets with $N = 2000$, they are 369.6 ms and 44.6 ms.

Table 14: Controlled inference-time comparison on an NVIDIA RTX 3090 using identical dataset shapes and repeated measurements. CoClusterPFN timing excludes PSM decoding to isolate neural inference. Times include input transfer and probability-output transfer but exclude preprocessing, checkpoint loading, and evaluation metrics. The ratio is computed per dataset before taking the median.

Dataset size	Datasets	CoClusterPFN mean	TabClustPFN mean	Median ratio
$N \leq 200$	11	21.3 ms	27.4 ms	$0.80\times$
$201 \leq N \leq 500$	9	27.2 ms	29.4 ms	$0.92\times$
$501 \leq N \leq 1000$	8	81.0 ms	31.2 ms	$2.48\times$
$1001 \leq N \leq 2000$	32	310.1 ms	41.9 ms	$7.67\times$
All datasets	60	184.2 ms	35.9 ms	$3.48\times$

Observed classical-pipeline runtimes. For context, we additionally report the wall-clock times recorded during the complete CLUBench evaluation. These measurements represent single end-to-end executions rather than the repeated, shape-matched protocol used in Table 14. They should therefore be interpreted as computational-footprint estimates, not controlled latency comparisons. Table 15 reports these measurements for the classical pipelines, including the fitting and selection operations named in the table; each pipeline returns a hard partition.

Table 15: Observed end-to-end runtimes for the classical pipelines during the CLUBench evaluation. Values are single-pass pipeline records and are not controlled latency comparisons.

Method	Median/dataset	Mean/dataset
Ward	30.8 ms	49.4 ms
k-means++	64.2 ms	118.8 ms
Agglomerative + silhouette linkage selection	106.7 ms	172.2 ms
GMM + BIC covariance selection	246.0 ms	393.0 ms
Spectral + neighbor selection	516.4 ms	683.3 ms

The k-means++ mean includes one anomalous 3.38-second measurement on a small dataset. Its mean after removing that observation is 63.5 ms, close to its 64.2 ms median. We report the recorded value in the table.

C.7 METRIC-ALIGNED COMPARISONS

Table 16 compares the synthetic-validation-selected CoClusterPFN decoder for each metric with TabClustPFN. Positive differences favor CoClusterPFN for ARI and NMI. For VI and N-invariant Binder loss, the table reports the positive reduction in loss, so larger values again favor CoClusterPFN.

All paired intervals exclude zero in favor of CoClusterPFN except the supplied-K Binder-loss interval, which includes zero.

Table 16: Paired metric-aligned differences between CoClusterPFN and TabClustPFN over 60 CLUBench datasets. For VI and Binder, improvement denotes the reduction in loss. Binder reductions use the N-invariant loss in Equation (31).

Setting	Metric	Improvement	Paired 95% interval
Known K	ARI	$+0.045237$	$[0.007926, 0.085880]$
Known K	NMI	$+0.046575$	$[0.014124, 0.084597]$
Known K	VI reduction	0.161971	$[0.097904, 0.228702]$
Known K	Binder reduction	0.016524	$[-0.003876, 0.038307]$
Unknown K	ARI	$+0.045770$	$[0.019320, 0.074388]$
Unknown K	NMI	$+0.049142$	$[0.024670, 0.076221]$
Unknown K	VI reduction	0.307796	$[0.201303, 0.415972]$
Unknown K	Binder reduction	0.032564	$[0.005830, 0.060111]$

D IRIS MULTIRESOLUTION CASE STUDY

Iris is one of the 60 datasets in the frozen CLUBench benchmark. We apply the same preprocessing and CoClusterPFN checkpoint as in the primary evaluation and provide the unknown-K code, so the predicted PSM marginalizes over the supported counts $K \in \{2, \dots, 10\}$ under the training prior. Neither the reference count nor the pointwise species annotations are used to construct the PSM or the hierarchy.

Figure 3 organizes the pairwise dissimilarities $1 - \widehat{S}_{ij}$ using complete linkage. MaxPEAR selects $K = 3$. We use the complete-linkage tree to display cuts at $K \in \{2, 3, 4, 5\}$. The aligned color strips above the PSM show how each successive cut divides the dendrogram-ordered observations, while the scatterplots display the corresponding partitions in the first two principal components. PCA is used only for visualization; CoClusterPFN receives all four standardized Iris features.

For the baseline comparison, panels (b), (g), and (h) of Figure 3 display CoClusterPFN, TabClustPFN, and GMM+BIC in the same CoClusterPFN dendrogram order. TabClustPFN uses its frozen unknown-count pipeline and predicts $\widehat{K} = 4$, while GMM+BIC jointly selects its covariance family and $\widehat{K} = 2$. For TabClustPFN and GMM+BIC, the displayed similarity matrices are constructed from their responsibilities using Equation 4. Among the $\binom{150}{2}$ unordered off-diagonal pairs, the fractions with similarities in $(0.1, 0.9)$ are 43.75% for CoClusterPFN, 0.69% for TabClustPFN, and 0.00% for GMM+BIC. Reference species labels are not used for count selection, matrix construction, or observation ordering.

The PSM contains one sharply separated 50-observation block: its within-block probabilities are high, its probabilities with the remaining observations are close to zero, and this group remains intact at every displayed resolution. At $K = 2$, the other 100 observations form one group. The $K = 3$ cut divides this less certain region into groups of sizes 53 and 47, producing cluster sizes $50/53/47$ and an ARI of 0.8681 against the Iris species annotations.

Finer cuts act locally rather than reorganizing the entire partition. The $K = 4$ cut divides the 53-observation branch into groups of sizes 34 and 19, whereas the $K = 5$ cut divides the other 47-observation branch into groups of sizes 29 and 18. The graded probabilities within the lower-right PSM block expose this weaker internal separation, whereas a single hard partition would show only one of these resolutions.

The dendrogram should not be interpreted as a posterior distribution over hierarchical trees, and its cuts are not posterior probabilities over K. It is a deterministic, post-hoc organization of the predicted pairwise probabilities. The case study instead demonstrates how one marginalized PSM can expose both stable co-clustering structure and where alternative levels of partition granularity arise, while leaving the final resolution to a separate decision rule.

E ADDITIONAL UNKNOWN-COUNT DIAGNOSTICS

This appendix gives a complete conditioning and decoding ablation complementing the CLUBench benchmark in Section 4.3. It uses the same 60 datasets, label-independent uniform subsamples, preprocessing, checkpoint, and predicted PSMs as the frozen benchmark.

Three-mode diagnostic. MaxPEAR is the prespecified decoder for ARI, selected on 256 synthetic validation tasks without CLUBench labels. The validation collection contains 64 tasks from each of the four training observation-count bands. We compare three conditions. First, CoClusterPFN receives the known-K code and MaxPEAR is constrained to the reference count. Second, the model receives the unknown-K code while MaxPEAR remains constrained to that count; comparing these rows isolates the effect of withholding K from the learned PSM. Third, the same unknown-code PSM is decoded by free-K MaxPEAR over candidates with two to ten groups; comparing the final two rows isolates the additional effect of withholding K from the decoder.

We report means across datasets and 95% percentile intervals from 20,000 paired dataset-level bootstrap resamples drawn with replacement (Efron & Tibshirani, 1993). Table 17 reports the three conditions.

Table 17: Cluster-count conditioning ablation on the frozen, label-independent CLUBench evaluation sample. All partitions use MaxPEAR, the synthetic-validation-selected decoder for ARI in both count settings.

Condition	ARI	NMI	Pair BCE
Known code, decoder given K	0.389543	0.399038	0.607402
Unknown code, decoder given K	0.383858	0.394196	0.743453
Unknown code, decoder not given K	0.325941	0.362138	0.743453

With the decoder constrained to the reference count, withholding K from the model changes ARI by -0.005685 (unknown code minus known code; paired 95% interval $[-0.021273, 0.012890]$). Pairwise BCE increases by 0.136051, showing that the count code has a clearer effect on the probability matrix than on the resulting reference-count partition.

Removing the reference count from the decoder changes ARI by -0.057916 (free-K minus constrained-K; paired 95% interval $[-0.105906, -0.018854]$). Pairwise BCE is unchanged because both rows decode the same unknown-code PSM.

F MATCHED-PRIOR POSTERIOR AGREEMENT AND CALIBRATION

PSM mean squared error. For one dataset, PSM MSE is the mean squared difference between PFN and MCMC pair probabilities, excluding diagonal entries:

$$\mathrm{MSE}_{\mathrm{PSM}}(\widehat{S}, S^{\mathrm{MCMC}}) = \frac{1}{\binom{N}{2}} \sum_{i<j} \left(\widehat{S}_{ij} - S^{\mathrm{MCMC}}_{ij}\right)^2. \tag{32}$$

Table 19 provides the complete metrics for the mixed-K experiment summarized in Section 4.1. Table 18 reports a separate fixed-K matched-prior experiment used for the controlled timing comparison later in this appendix.

Table 18: Agreement between the CoClusterPFN PSM and a four-chain collapsed-Gibbs reference in the separate fixed-K experiment used for the controlled timing comparison. Results are computed over 50 held-out datasets from the matched Bayesian GMM. Excess CE is the cross-entropy above the entropy of the MCMC target itself.

PSM MAE ↓	PSM RMSE ↓	Correlation ↑	Excess CE ↓
0.0284	0.0357	0.9776	0.0040

Table 19: Conditional and marginalized PSM prediction under the mixed-K prior. Both rows use the same 45 datasets passing the reference diagnostics in both modes; each prediction is evaluated against its corresponding MCMC posterior target. PSM MSE is the mean squared difference between PFN and MCMC probabilities over unordered off-diagonal pairs, averaged over datasets.

PFN mode	PSM MAE ↓	PSM MSE ↓	Excess CE ↓	Correlation ↑
Known K	0.0187	0.00089	0.00251	0.9800
Unknown K	0.0302	0.00161	0.00420	0.9708

Separate fixed-K experiment. The fixed-K experiment used for the controlled timing comparison has $N = 48$ observations, $d = 2$ features, and three component slots. The mixture weights and component parameters follow

$$\pi \sim \mathrm{Dirichlet}(2, 2, 2), \tag{33}$$

$$\Sigma_c \sim \mathrm{IW}(7, I_2), \tag{34}$$

$$\mu_c \mid \Sigma_c \sim \mathcal{N}(0, \Sigma_c/1.5), \qquad c = 1, 2, 3. \tag{35}$$

Here, IW denotes the inverse-Wishart distribution with seven degrees of freedom and scale matrix I_2, the 2×2 identity matrix. Observations are generated as

$$z_i \sim \mathrm{Categorical}(\pi), \tag{36}$$

$$x_i \mid z_i = c, \mu_c, \Sigma_c \sim \mathcal{N}(\mu_c, \Sigma_c), \qquad i = 1, \ldots, 48. \tag{37}$$

Training and held-out datasets use this same prior, without normalization or rejection filtering.

Both PFN and MCMC condition on $K = 3$ in this experiment.

Mixed-K prior and training. The extension uses $N = 48$, $d = 2$, and the same component priors, but places a uniform prior on the total number of component slots:

$$K \sim \mathrm{Uniform}\{2, \ldots, 10\}, \tag{38}$$

$$\pi \mid K \sim \mathrm{Dirichlet}(\underbrace{2, \ldots, 2}_{K \text{ entries}}). \tag{39}$$

For each $c = 1, \ldots, K$, the covariance and mean priors remain $\Sigma_c \sim \mathrm{IW}(7, I_2)$ and $\mu_c \mid \Sigma_c \sim \mathcal{N}(0, \Sigma_c/1.5)$. Assignments and observations are generated as in the fixed-K experiment. Thus, in this appendix K denotes total mixture components, whereas the tabular benchmark uses the number

of occupied reference classes. Both the fixed-K and mixed-K PFNs use the same architecture, a batch size of 64, and 30,000 training updates. The fixed-K model optimizes only known-K prediction. The mixed-K PFN is trained with K sampled uniformly from this support, and each task contributes equally weighted pairwise losses for the conditional probabilities $P(z_i = z_j \mid X, K)$ and the marginalized probabilities $P(z_i = z_j \mid X)$. Neither normalization nor dataset rejection is applied to the generative process.

The mixed-K checkpoint has training seed 1702.

Checkpoint SHA-256:
`0c0373e299a4e12150c6abf052a0051b0a6bc463b0ef0a58dc57e87e943d6573`

The checkpoint was not changed for the longer MCMC rerun: its predictions are bitwise identical to those used in the earlier evaluation on all 21 datasets qualified by both versions.

Posterior targets and evaluation design. The known-K reference conditions on the generating number of component slots. The unknown-K reference instead targets

$$P(z_i = z_j \mid X) = \sum_{k=2}^{10} P(z_i = z_j \mid X, K = k) P(K = k \mid X). \tag{40}$$

Its sampler uses the corresponding mixture-of-finite-mixtures representation, integrating out empty components. The total component count K is distinct from the number of occupied components, which can be smaller.

The evaluation collection is balanced over generating K: six datasets are sampled conditional on each $K \in \{2, \ldots, 10\}$, for 54 datasets in total. This design gives equal coverage of the support; unknown-K posterior inference still uses the uniform prior over total component count. References are screened using the prespecified diagnostics described below. All 54 known-K references and 45 unknown-K references pass. The reported mixed-K comparison uses the same 45 datasets qualifying in both modes. The qualified subset includes four or more datasets at every generating count from two to ten. This reference-quality screening is separate from the generative process and does not use PFN prediction errors.

Dataset index $i \in \{0, \ldots, 53\}$ uses seed $91001 + 1009i$ and generating count $2 + (i \bmod 9)$. For chain $c \in \{1, \ldots, 4\}$, the known-K and unknown-K seeds are respectively the dataset seed plus $100003c$ and $200003c$. The known-K sampler starts from independent uniform assignments over the supplied component slots. The unknown-K sampler first draws a total component count uniformly from two to ten, initializes assignments uniformly, and removes empty slots. Both references use sequential single-site collapsed Gibbs updates, with no split–merge moves.

MCMC settings and reference diagnostics. Table 20 distinguishes the original fixed-K sampling budget from the longer mixed-K runs. In the mixed-K experiment, the listed settings apply separately to each dataset's known-K and unknown-K reference, giving 32,000 retained partitions per reference.

Table 20: MCMC settings for the separate fixed-K experiment and the mixed-K extension. The mixed-K settings apply to both conditional and marginalized reference inference.

Setting	Fixed-K experiment	Mixed-K experiment
Chains	4	4
Burn-in sweeps per chain	1,000	4,000
Retained draws per chain	1,000	8,000
Thinning interval	2	1
Total retained partitions	4,000	32,000

The separate fixed-K study monitors ordinary split-$\widehat{R}$ for co-clustering pair mass and occupied-component count, together with disagreement between chain-specific PSMs. Pair mass is the fraction of unordered observation pairs assigned to the same occupied component in a sampled partition. The mixed-K study additionally uses the rank-normalized folded split-$\widehat{R}$ and bulk effective sample size (ESS) for both quantities. A mixed-K reference is accepted only if

$$\widehat{R}_{\mathrm{rank}} \leq 1.01 \qquad \text{and} \qquad \mathrm{ESS}_{\mathrm{bulk}} \geq 400 \tag{41}$$

for both pair mass and occupied-component count. These checks are applied separately to the known-K and unknown-K chains.

Known-K inference passes all four criteria on all 54 datasets. Unknown-K inference passes on 45 datasets. Among the nine failures, seven datasets fail at least one $\widehat{R}$ criterion and two different datasets fail only the occupied-count ESS criterion; no dataset fails pair-mass ESS. The excluded zero-based dataset indices are $0, 1, 10, 22, 23, 27, 31, 32$, and 47. Table 21 reports qualification by generating count, and Table 22 summarizes the diagnostic ranges.

Table 21: Unknown-K reference qualification by generating count under the longer mixed-K MCMC run.

Generating K	2	3	4	5	6	7	8	9	10
Passed (of 6)	4	4	5	6	4	4	6	6	6

Table 22: Minimum, median, and maximum mixed-K reference diagnostics over all 54 datasets. The displayed $\widehat{R}$ is the gating rank-normalized folded split statistic.

Mode	Diagnostic	Minimum	Median	Maximum
Known K	Pair-mass ESS	3997.3	10900.6	17649.0
Known K	Occupied-count ESS	15645.6	23311.0	32000.0
Unknown K	Pair-mass ESS	411.1	812.0	2509.8
Unknown K	Occupied-count ESS	307.8	721.8	2313.2
Known K	Pair-mass $\widehat{R}$	0.99999	1.00025	1.00089
Known K	Occupied-count $\widehat{R}$	0.99992	1.00004	1.00026
Unknown K	Pair-mass $\widehat{R}$	1.00060	1.00379	1.01212
Unknown K	Occupied-count $\widehat{R}$	1.00060	1.00433	1.01410

Chain-specific PSMs are also compared with their pooled four-chain reference. Averaged over all 54 datasets, chain-to-pooled PSM MAE is 0.00345 for known K and 0.00715 for unknown K; the corresponding mean worst-chain RMSEs are 0.00479 and 0.01185. The mean maximum range across chain estimates of an individual pair probability is 0.02923 and 0.04679, respectively. These summaries diagnose chain agreement but do not establish convergence of every individual PSM entry.

Conditional–marginalized reference-PSM difference. The reported difference between conditional and marginalized reference PSMs is their mean absolute difference on those same 45 datasets. With d indexing datasets, it is

$$\frac{1}{45\binom{48}{2}} \sum_{d=1}^{45} \sum_{i<j} \left| S_{ij,d}^{\mathrm{MCMC,unknown}} - S_{ij,d}^{\mathrm{MCMC,known}} \right| = 0.0848. \tag{42}$$

Diagonal entries are excluded. This quantity measures how marginalizing over K changes the reference posterior pair probabilities, not PFN prediction error. Its population standard deviation across the 45 datasets is 0.0667. For comparison, the mean is 0.0973 over all 54 datasets, including the nine unknown-K references that fail the prespecified diagnostics.

Dataset-level variability. Table 23 reports population standard deviations over these 45 datasets qualifying in both modes and percentile intervals from dataset-level bootstrap resampling.

Fixed-K matched-target timing. The fixed-K timing comparison measures the same posterior-similarity target for CoClusterPFN and the collapsed Gibbs reference. Model loading is excluded, GPU operations are synchronized, and the four CPU NumPy MCMC chains are executed sequentially. After a CUDA warm-up, CoClusterPFN averages 0.001863 seconds per dataset and MCMC averages 90.710 seconds, giving a ratio of approximately $49{,}000\times$. If the first CUDA warm-up is charged to the PFN, the ratio is approximately $6{,}450\times$.

Table 23: Across-dataset standard deviations and dataset-bootstrap 95% intervals for mean PFN agreement with the corresponding MCMC target on the 45 datasets qualifying in both modes.

Mode	Metric	Across-dataset SD	95% interval for mean
Known K	PSM MAE	0.01377	[0.01504, 0.02283]
Known K	PSM MSE	0.00156	[0.00050, 0.00138]
Known K	Excess CE	0.00430	[0.00143, 0.00398]
Known K	Correlation	0.01642	[0.97499, 0.98452]
Unknown K	PSM MAE	0.01410	[0.02615, 0.03430]
Unknown K	PSM MSE	0.00148	[0.00122, 0.00209]
Unknown K	Excess CE	0.00397	[0.00319, 0.00554]
Unknown K	Correlation	0.02044	[0.96446, 0.97645]

Reference-computation time. The high-budget MCMC references were computed using six concurrent single-CPU jobs, each processing nine datasets sequentially. Over all 54 datasets, known-K reference computation averaged 692.0 seconds per dataset (median 693.5; range 329.8–1060.7), while unknown-K computation averaged 741.6 seconds (median 756.5; range 475.2–946.0).

G MATCHED-PRIOR AMORTIZED-COMPARISON DETAILS

Common task distribution. All three model families were trained on the task distribution described in Section 4.1. The total component count is sampled uniformly from $\{2, \ldots, 10\}$. A task first selects one of the four observation-count bands 10–25, 26–50, 51–100, and 101–200 with equal probability, then samples N uniformly within that band. Component weights have a symmetric Dirichlet prior with concentration 2 per component, with the same covariance and mean priors as the mixed-K experiment in Appendix F. Empty sampled components are permitted, so the reference count is the number of occupied components. No centering, standardization, or coordinate rescaling is applied during training or evaluation.

CoClusterPFN and DAC receive independently generated tasks within each batch. Following the NCP training interface, an NCP update uses one sampled partition and independently sampled NIW–GMM parameters for the batch members conditional on that partition; each individual dataset has the same marginal prior.

The test collection contains 2,500 independently generated datasets, with 500 datasets at each $N \in \{10, 25, 50, 100, 200\}$. The paired intervals in Section 4.1 use 20,000 dataset-level bootstrap resamples.

Decoder selection. For CoClusterPFN and NCP separately, we select among MaxPEAR, MinBinder, and MinVI-LB using 256 additional matched-prior validation tasks generated with seed 20260924, with 64 tasks drawn from each observation-count band. Both methods select MaxPEAR for ARI, MinBinder for NMI and N-invariant Binder loss, and MinVI-LB for VI. Count diagnostics use the ARI-selected MaxPEAR partitions. The 2,500 test tasks are not used for decoder selection.

Additional result diagnostics. CoClusterPFN's paired ARI advantage is 0.0546 over NCP, with a 95% paired bootstrap interval of $[0.0499, 0.0594]$; 0.0718 over anchored DAC, with interval $[0.0668, 0.0768]$; and 0.1691 over minimum-loss DAC, with interval $[0.1627, 0.1753]$. The corresponding wins/ties/losses are $1844/36/620$, $1896/26/578$, and $2258/23/219$ across the 2,500 test tasks.

The ARI advantage occurs at every evaluated observation count. At $N = 10$ and $N = 200$, respectively, mean CoClusterPFN ARI is 0.1695 and 0.2010, compared with 0.1289 and 0.1334 for NCP and 0.0981 and 0.1288 for anchored DAC.

The remaining metrics expose different behaviors. Anchored DAC obtains the highest NMI but extracts many small groups, yielding count MAE 17.77 despite a prior with at most ten components. Minimum-loss DAC frequently returns its fallback group and performs substantially worse in ARI, NMI, and Binder loss. CoClusterPFN obtains the best exact-count accuracy and count MAE, at 0.1548 and 2.3044, narrowly ahead of NCP at 0.1356 and 2.3304.

CoClusterPFN training and inference. The matched CoClusterPFN has two Transformer layers, hidden width 64, four attention heads, and a direct pairwise head. It was optimized for 30,000 steps with batches of 64 tasks, AdamW, peak learning rate 3×10^{-4}, 1,500 warm-up steps followed by cosine decay, weight decay 10^{-4}, and gradient-norm clipping at 1. Training assigns equal weight to the known- and unknown-count pairwise BCE objectives.

Checkpoints were evaluated every 250 steps on a fixed collection of 256 synthetic tasks, with 64 tasks from each observation-count band. The selected checkpoint minimizes the mean of the known- and unknown-count validation BCE and occurs at step 24,500. Unknown-count inference uses the unknown-count code and constructs the complete PSM in one network evaluation. The independently selected metric-specific decoder is then applied over candidate partitions with at most ten groups.

NCP training and inference. NCP was optimized for 10,000 iterations with batch size 16, one random point ordering per update, Adam with initial learning rate 10^{-4} and weight decay 0.01, and training seed 42. The learning rate was reduced to 5×10^{-5} after 3,500 iterations and to 10^{-5} after 6,500 iterations. Evaluation uses the final checkpoint. For each test task, NCP draws ten autoregressive partitions. Their co-membership matrices are averaged to form an empirical PSM. The independently selected metric-specific decoder is then applied with candidate counts from two through ten.

DAC training and inference. We trained separate minimum-loss and anchored-filtering DAC models for 20,000 steps with batch size 100, Adam with initial learning rate 5×10^{-4}, cosine decay, gradient-norm clipping at 10, and training seed 123. Both use the Gaussian density branch and their respective published filtering objectives. Evaluation follows the released implementation: membership logits above zero, corresponding to probabilities above 0.5, define the cluster extracted at each pass. Filtering continues until all observations are assigned or 50 passes have been performed.

DAC filtering failures are retained in the reported scores rather than discarded. Anchored filtering assigns every observation on 86.8% of tasks; it leaves a mean of 6.61 observations unassigned and a maximum of 145. Minimum-loss filtering assigns every observation on only 4.04% of tasks, leaving a mean of 72.41 observations unassigned and a maximum of 200. As in the released iterative routine, observations remaining after filtering receive the initial fallback label.

A diagnostic on a separate collection of 250 synthetic tasks found that the networks preserve ranking signal, but the standard threshold selects only 2.3 observations for a minimum-loss target cluster of mean size 15.4, and 8.6 observations for an anchored target cluster of mean size 21.3. We use the published threshold because changing it only for DAC would introduce method-specific inference calibration.

Table 24 records the exact checkpoint hashes used in this comparison.

Table 24: Frozen checkpoints for the matched-prior amortized comparison.

Model	SHA-256
CoClusterPFN	`1c49b208a7868f9d163e3d06436fc0f5123375916acf0e96f86868837bfba957`
NCP	`5a72b3c154d16404b6310ea5167d7fc04c1d3620638b3e1a1074fac4d30a721d`
DAC, minimum-loss filtering	`a46171fd5b7eaafbf4b8295ab77fd04e77a01da7b4ec65ebc2e5d049e8484af2`
DAC, anchored filtering	`7ea7285cce8e9d8feb0de843964fce963f06894847d05751f50998cecfaefd97`