TabClustPFN: A Prior-data Fitted Network for Tabular Data Clustering

Tianqi Zhao
Renmin University of China

Guanyang Wang
Rutgers University

Yan Shuo Tan
National University of Singapore

Qiong Zhang*
Mohamed bin Zayed University
of Artificial Intelligence

Abstract

Prior-data Fitted Networks (PFNs) have reframed supervised tabular learning as single-pass in-context inference without per-dataset optimization. Extending this paradigm to clustering is appealing yet fundamentally more challenging, due to absent supervision, unknown cluster cardinality, and label switching inherent to partition outputs. Existing PFN-based clustering methods address these challenges only partially, either requiring known cardinality as input or relying on unstable label-ordering conventions and overly restrictive synthetic priors. We introduce `TabClustPFN`, a clustering PFN that resolves all challenges jointly through co-designed prior, objective, and architecture. Our hybrid pretraining prior captures heterogeneous real-tabular geometry; our decoupled partition inference network–cardinality inference network architecture jointly infers cluster assignments and cardinality in a single pass; and our SoftARI training objective is permutation-invariant by construction, eliminating the need for any label-ordering convention. On a 44 curated real-world tabular benchmark, `TabClustPFN` achieves state-of-the-art clustering performance against classical, deep, and amortized baselines, with runtime comparable to efficient baselines. Code is available here.

1 Introduction

Prior-data Fitted Networks (PFNs) [12, 28, 11, 21, 33, 29] have established a new paradigm for supervised tabular learning. By pretraining a transformer to approximate a Bayesian posterior predictive distribution over a broad synthetic prior, PFN-based models such as TabPFN v2.5 [11] achieve strong generalization via in-context-learning in a single forward pass, without any per-dataset optimization. This approach has proven remarkably effective, delivering state-of-the-art performance [7] across diverse tabular classification and regression tasks while remaining orders of magnitude faster than dataset-specific alternatives.

Applying this paradigm to clustering would be highly valuable. While clustering is central to tabular exploratory analysis, the lack of labels forces costly

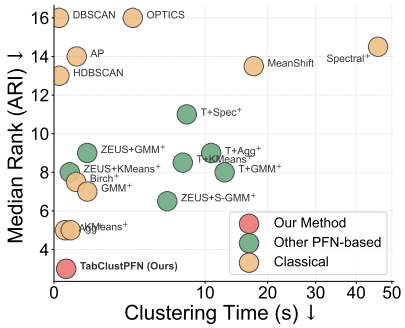

Figure 1: **Time & median ARI rank (lower is better).** `TabClustPFN` achieves the best ARI with runtime comparable to simple methods and faster than other PFN-based methods.

*Correspondence: `qiong.zhang@mbzuai.ac.ae`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

trial-and-error and unreliable heuristics to select the cardinality, an algorithm with a suitable inductive bias, and hyperparameters. The PFN paradigm offers a way to amortize these choices into pretraining, using a synthetic prior to encode clustering biases and in-context inference to predict both $\boldsymbol{Z}$ and K in a single pass. However, employing the PFN paradigm to clustering is fundamentally harder than its supervised counterpart.

Three obstacles arise specifically from the unsupervised nature of clustering:

(1) No supervision signal: unlike supervised PFNs, where labels define class boundaries, clustering contexts contain only features. The model must hence infer the notion of clusters from a synthetic prior, making its design critical to capturing meaningful cluster structure and real-data geometry.

(2) Unknown cardinality: Context labels define the output space in supervised PFNs, whereas clustering must infer K from data, a challenge even for classical methods [20, 8].

(3) Label switching: any permutation of the cluster labels yields an equally valid partition. Standard cross-entropy objectives assume a fixed label correspondence and are therefore inapplicable.

Recent PFN-based methods *only partially* address these challenges. ZEUS [22] learns expressive tabular embeddings but delegates clustering to k-means with known K, hence leaving challenge (2) unresolved. Cluster-PFN [3] jointly infers assignments and cardinality but resolves label switching by hard ordering clusters according to their distance to the origin, a convention that is undefined when two centers are equidistant (*e.g.,* -1 and $+1$ in 1D) and unstable in high dimensions [1]. Its pretraining prior is further restricted to GMMs with $D \leq 5$, which do not capture the nonlinear geometry or feature heterogeneity of real tabular data. See a detailed comparison in App. D.4.

No existing method resolves all 3 challenges simultaneously. We introduce `TabClustPFN`, a PFN that jointly resolves these challenges via a tightly co-designed prior, objective, and architecture. This joint design is crucial: realistic tabular priors induce diverse partitions that demand permutation-invariant loss, while robust inference of K under such partition-level objectives requires decoupling cluster-assignment inference from cardinality inference. Our contributions are:

- **A hybrid pretraining prior [challenge (1)].** We combine controlled-overlap GMMs with nonlinear invertible transformations and mixed numerical-categorical feature generation [22], covering the heterogeneous geometries and feature types found in real tabular data. Ablations confirm each component contributes to generalization (Tab. 7).
- **A decoupled novel PIN-CIN architecture [challenge (2)].** Our model has 2 components: a Partition Inference Network (PIN) maintains K explicit cluster prototypes, iteratively co-refined with data representations via cross-attention; and a Cardinality Inference Network (CIN) estimates $p(K|\boldsymbol{X})$ from permutation-invariant Gram matrix summaries of PIN outputs across all candidate K simultaneously, unlike prior methods that require K as input or use sequential single K extraction.
- **SoftARI: a permutation-invariant training objective [challenge (3)].** We train the PIN with a differentiable relaxation of the Adjusted Rand Index (ARI), which is permutation-invariant by construction and requires no label ordering convention. SoftARI matches or outperforms matching-based cross-entropy alternatives while halving per-iteration pretraining cost (Tab. 5).
- **State-of-the-art results on a broad tabular clustering benchmark.** We introduce a curated benchmark of 44 real-world tabular datasets and evaluate `TabClustPFN` against classical, deep, and amortized clustering baselines (App. A). `TabClustPFN` achieves SOTA performance (Fig. 1), confirmed by paired t-tests, Wilcoxon tests, and critical-difference diagrams (App.D.6).

2 An overview of TabPFN

TabPFN [12] is a canonical *Prior-data Fitted Network* (PFN) that serves as a tabular foundation model for supervised learning, casting prediction as Bayesian inference. Given a dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ drawn IID from an unknown distribution π^* and a prior p over a space of data-generating distributions Π, TabPFN approximates the posterior predictive distribution (PPD): $p(y|x, \mathcal{D}) = \int_\Pi \pi(y|x)\, p(\pi|\mathcal{D})\, d\pi$. To achieve this without intractable integrals, TabPFN uses PFN: a transformer is pretrained to approximate this PPD directly, enabling in-context learning (ICL). At test time, the model parameters are fixed; it processes the dataset $\mathcal{D}$ as a sequence of tokens in the context window and predicts the target y for a query x in a single forward pass. Another core innovation of TabPFN lies in its pretraining data. Instead of using real-world datasets, the model is pretrained on a

large collection of synthetic data sampled from a prior defined by structural causal models [27]. This prior is designed to produce distributions that mimic the complexity of real tabular data, including nonlinear feature interactions, heteroskedastic noise, and mixed feature types. Pretraining proceeds by repeatedly sampling synthetic datasets from this prior ($\approx$ 130M datasets). Each dataset is divided into a context set (pseudo-training) and a query set (pseudo-test), and the transformer is trained to maximize the likelihood of query labels given the context.

In summary, the PFN framework comprises three components: (i) a ***task-level prior*** that defines the distribution over data-generating processes; (ii) a ***transformer architecture*** that performs amortized inference via ICL; and (iii) a ***pretraining objective*** that aligns predictions with the true PPD. This structure provides a principled blueprint for extension.

3 TabClustPFN

We consider a latent-variable model where each observation X_i is generated independently from one of K clusters,

$$X_i | Z_i = k \sim f_k, \tag{1}$$

with latent assignments $Z_i \in [K] = \{1, \dots, K\}$ and unknown cluster distributions f_k for $K \geq 2$. Given a dataset $\boldsymbol{X} = \{X_i\}_{i=1}^N$, we perform Bayesian inference over both assignments $\boldsymbol{Z} = \{Z_i\}_{i=1}^N$ and the cluster number K, approximating the joint posterior

$$p(K, \boldsymbol{Z} | \boldsymbol{X}) = p(\boldsymbol{Z} | K, \boldsymbol{X})\, p(K | \boldsymbol{X}),$$

which factorizes into a posterior over partitions conditioned on the cluster number K and a marginal posterior over K. `TabClustPFN` mirrors this factorization with two modules. The *Partition Inference Network* (PIN) approximates $p(\boldsymbol{Z} | K, \boldsymbol{X})$ and performs amortized inference over cluster assignments for a given K. The *Cardinality Inference Network* (CIN) approximates $p(K | \boldsymbol{X})$ and predicts cluster number directly from the dataset. Both components are implemented as prior-fitted networks that operate via in-context learning, enabling dataset adaptation without retraining. Fig. 2 overviews the architecture and loss functions of `TabClustPFN`, while Sec. 5 provides ablations of the design choices. We describe each of the three components of our PFN-based method respectively below. All details related to pretraining are given in App. B.

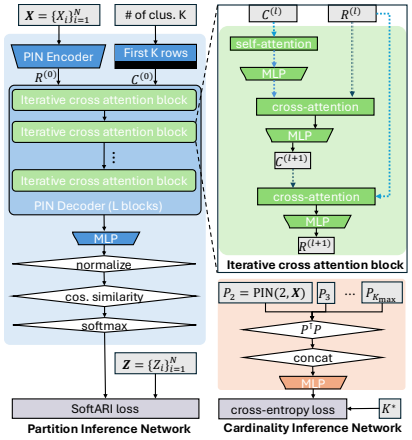

Figure 2: **TabClustPFN architecture & loss.** Given dataset $\boldsymbol{X}$ and candidate cluster number K, the PIN uses a transformer encoder followed by L layers of iterative cross-attention between data representations and learnable cluster prototypes to produce soft assignments P. Dotted blue lines denote query paths, and dashed arrows indicate keys and values. Cosine similarity between normalized representations and prototypes yields cluster posteriors. For $K \in \{2, \dots, K_{\max}\}$, Gram matrices $P_K^\top P_K$ are summarized and fed to an MLP-based CIN to estimate $p(K | \boldsymbol{X})$. PIN is trained with permutation-invariant SoftARI loss, while CIN uses cross-entropy on true K^* under decoupled optimization.

3.1 Prior data

The effectiveness of a PFN depends critically on the diversity and fidelity of its pretraining prior. Accordingly, we design a synthetic data-generating process (DGP) that follows (1), where each dataset is generated from latent cluster assignments Z_i and cluster-specific distributions f_{Z_i} (see App. B.1). To ensure rich and challenging training, we define two complementary classes of cluster distributions f_k. In the first class, we consider *GMM-based clusters*. Each f_k is a Gaussian, forming a standard GMM. To vary the difficulty of clustering across datasets, we explicitly control the degree of cluster overlap. For each dataset, we sample a target maximum pairwise overlap, measured via Bayes misclassification probability, so that some datasets contain well-separated clusters while others contain highly overlapping, more challenging clusters. Further diversity is achieved by randomizing mixture proportions and covariance structures, including spherical versus ellipsoidal shapes and homoscedastic versus heteroscedastic settings. In the second class, we consider *iResNet-transformed clusters*. We generate K latent Gaussian clusters and then apply a randomly sampled invertible iResNet [2] transformation following ZEUS [22]. This bijective, nonlinear mapping

produces complex, non-Gaussian cluster shapes while preserving cluster identity and the number of components. Categorical variables are incorporated following the same ZEUS procedure. We call this the *ZEUS prior* in short. Together, these two mechanisms ensure that pretraining exposes the PFN to both standard parametric clusters and more complex, highly structured cluster geometries.

3.2 Network architecture

Transformer based PIN. We implement PIN as a Transformer based PFN mapping a dataset $\boldsymbol{X}$ and cluster cardinality K to a soft partition $\text{PIN}_\theta : (\boldsymbol{X}, K) \mapsto P \in [0,1]^{N\times K}$, $P_{ik} = \hat{p}(Z_i = k|K, \boldsymbol{X})$. PIN follows an encoder–decoder design: the encoder maps $\boldsymbol{X}$ to representations in $\mathbb{R}^d$, while the decoder refines them using learnable cluster prototypes. For each K, the decoder selects the first K prototypes from a shared $K_{\max} \times d$ matrix and applies L layers of prototype self-attention followed by bidirectional cross-attention between prototypes and data, enabling iterative co-refinement. *Crucially, both the class prototypes and the input samples are permutation-invariant with respect to their row ordering.* Motivated by Set Transformer [17], all attention blocks in the PIN decoder are therefore implemented without positional encodings. Finally, normalized data representations and prototypes are compared via cosine similarity, and a softmax over clusters yields assignment probabilities. This realizes a fully amortized, permutation-invariant analogue of centroid-based clustering. We adopt the TabICL encoder [28] (column-wise embedding followed by row-wise interaction) for reproducibility, though the framework is agnostic to the specific tabular encoder.

MLP based CIN. Rather than operating on the raw dataset, CIN leverages the partition structure inferred by PIN across candidate values of K. For each $K \in \{2, \dots, K_{\max}\}$, we compute the corresponding soft assignment matrix $P_K = \text{PIN}_\theta(\boldsymbol{X}, K)$ and form the Gram matrix $G_K = P_K^\top P_K$, which summarizes pairwise cluster co-assignment statistics. The lower-triangular entries of G_K are vectorized and concatenated across all candidate K, forming the input to a two-layer perceptron CIN_ϕ. To ensure invariance to permutations of cluster labels, we impose a canonical ordering: the diagonal elements and off-diagonal elements are sorted separately in descending order. This separation is necessary because the two groups encode fundamentally different quantities — diagonal entries correspond to cluster self-mass (soft cluster sizes), whereas off-diagonal entries represent inter-cluster co-assignment strength. Sorting them jointly would mix these semantically distinct statistics and distort their structural roles, while separate sorting preserves their meaning and removes dependence on arbitrary cluster index assignments. The network outputs a $(K_{\max} - 1)$-dimensional vector giving an estimated posterior over cluster cardinalities. We exclude $K = 1$, focusing on datasets that contain at least two latent clusters.

3.3 Loss functions

Loss for PIN. Clustering is invariant to permutations of cluster labels, so the training objective for PIN must not depend on their indexing. Standard supervised losses violate this requirement: identical partitions differing only by label permutations incur large loss, producing spurious signals and destabilizing training. A natural choice is the Adjusted Rand Index (ARI) [13], a widely used measure of agreement based on pairwise co-assignment. ARI is permutation-invariant and aligns with standard evaluation. However, it is defined over hard assignments via an $\arg\max$ and is therefore non-differentiable, making it incompatible with gradient-based optimization.

We use *SoftARI* [9], a differentiable relaxation that replaces hard assignments with the model's predicted cluster probabilities and computes soft pairwise co-assignment scores. Let $Z_{ik} = \mathbb{1}(Z_i = k)$ be the one-hot encoding of the ground-truth cluster assignment, which is available for synthetic pretraining datasets. SoftARI computes soft pairwise co-assignment scores using P_{ik} and Z_{ik}, yielding a fully differentiable and permutation-invariant objective:

$$\text{SoftARI}(\text{PIN}_\theta(K, \boldsymbol{X}), \boldsymbol{Z}) = \frac{\sum_{k,\ell} \binom{n_{k\ell}}{2} - \{\sum_k \binom{n_{k\cdot}}{2} \sum_\ell \binom{n_{\cdot\ell}}{2}\}/\binom{n}{2}}{\{\sum_k \binom{n_{k\cdot}}{2} + \sum_\ell \binom{n_{\cdot\ell}}{2}\}/2 - \{\sum_k \binom{n_{k\cdot}}{2} \sum_\ell \binom{n_{\cdot\ell}}{2}\}/\binom{n}{2}}$$

where $n_{k\ell} = \sum_{i=1}^n P_{ik} Z_{i\ell}$, $n_{k\cdot} = \sum_\ell n_{k\ell}$, and $n_{\cdot\ell} = \sum_k n_{k\ell}$. The loss for PIN is defined as

$$\mathcal{L}_{\text{PIN}}(\theta) = -\text{SoftARI}(\text{PIN}_\theta(K^*, \boldsymbol{X}), \boldsymbol{Z}), \tag{2}$$

where K^* is the true cluster number, available during pretraining on synthetic datasets (Sec. 3.1).

Loss for CIN. Since the true cluster number K^* is available from the DGP, we can supervise CIN using the standard cross-entropy loss over the candidate set $\{2, \dots, K_{\max}\}$. Let $\text{CIN}_\phi(\boldsymbol{X}) = \{\hat{p}(K|\boldsymbol{X})\}_{K=2}^{K_{\max}}$ denote the predicted posterior. The loss is

$$\mathcal{L}_{\text{CIN}}(\phi) = -\log \hat{p}(K^*|\boldsymbol{X}). \tag{3}$$

3.4 Model pretraining

Following the TabPFN paradigm, we pretrain the model on synthetic datasets sampled from the DGP described in Sec. 3.1. We initialize the encoder with pretrained TabICL weights [28] and randomly initialize the remaining parameters. At each optimization step, we draw a batch of 512 synthetic tasks (datasets). For each dataset, we sample the number of cluster $K \in [2, 10]$, the number of samples $N \in [500, 1000]$ and features $D \in [2, 64]$, sample a distribution over $\mathbb{R}^D$ from the prior, then generate the data and ground-truth cluster labels from this distribution. Each dataset is generated on-the-fly and never reused. We adopt a *decoupled optimization scheme* when optimizing the PIN loss in (2) and the CIN loss in (3). Specifically, CIN operates on detached PIN outputs, and its gradients are not backpropagated through PIN. This decoupling mitigates interference between partition inference and cardinality estimation, leading to improved performance (Tab. 6). All parameters are optimized with AdamW using a cosine learning-rate schedule with $2,000$ warm-up steps and a peak learning rate of 10^{-4}. Training runs for $10,000$ optimization steps on four RTX 5090 GPUs, yielding a favorable performance–compute trade-off at a cost of ≈ 92 GPU hours.

4 Empirical results

We evaluate the performance of our pretrained TabClustPFN on both unseen synthetic datasets and a curated real-world benchmark. We first describe the baseline methods and the evaluation metric.

Baselines. We evaluate `TabClustPFN` against a wide range of classical and modern clustering algorithms, covering scenarios with known and unknown cluster numbers. This includes all methods available in `scikit-learn` at the time of writing, as well as two PFN-based approaches: ZEUS+X [22] and T+X [19], where X denotes a classical clustering method used in an intermediate step. A full list of method names and their acronyms used in the results is provided in App. C.2. Methods marked with * indicate oracle knowledge of the cluster number, while those with $^+$ select it by maximizing the silhouette score. Methods without a suffix infer the number of clusters automatically. Since silhouette-based selection may not align with every method's clustering assumptions, we report additional results using method-specific criteria and hyperparameter search for DBSCAN, GMM, and Spectral (App. D.1). *We exclude Cluster-PFN [3] due to its different setting and instead compare it within its intended regime in App. D.4*. Cluster-PFN is not even comparable to GMM when $D \geq 3$. This results in 13 baselines with known K and 18 with selected K.

Table 1: **Performance on synthetic datasets**. Median ARI/NMI ranks and k-MAE (↓). Brackets show IQR. Bold and underline indicate best and second-best per column.

Method	ARI Rank [IQR] ↓	NMI Rank [IQR] ↓	k-MAE [IQR] ↓
TabClustPFN	**2 [4]**	**3 [5]**	**0 [1]**
AP	14 [1]	13 [3]	44 [12]
Agg$^+$	9 [4]	10 [4]	<u>1 [4]</u>
Birch$^+$	9 [3]	10 [3]	<u>1 [4]</u>
DBSCAN	16 [2]	17 [1]	5 [5]
GMM$^+$	6 [4]	6 [5]	<u>1 [3]</u>
HDBSCAN	15 [4]	15 [1]	3 [4]
KMeans$^+$	5 [5]	6 [4]	<u>1 [3]</u>
MeanShift	15 [2]	16 [2]	4 [5]
OPTICS	16 [2]	16 [1]	4 [6]
Spectral$^+$	14 [5]	14 [5]	2 [5]
T+Agg$^+$	5 [7]	5 [7]	<u>1 [3]</u>
T+GMM$^+$	5 [4]	<u>5 [5]</u>	<u>1 [3]</u>
T+KMeans$^+$	<u>4 [5]</u>	<u>5 [5]</u>	<u>1 [4]</u>
T+Spec$^+$	5 [5]	<u>5 [4]</u>	<u>1 [3]</u>
Zeus+GMM$^+$	10 [6]	9 [7]	6 [5]
Zeus+KMeans$^+$	9 [7]	9 [8]	6 [5]
Zeus+S-GMM$^+$	8 [6]	8 [7]	6 [5]

Performance metric. We assess clustering quality using standard metrics: ARI and Normalized Mutual Information (NMI). Because raw ARI and NMI values can vary substantially across datasets, we report rank-based metrics in the main text. Each method is ranked independently per dataset, and we report the median and IQR of the ranks. We use median (IQR) rather than mean (std) for robustness to outliers. When the cluster number is unknown, we additionally evaluate cluster cardinality estimation using the median absolute error (k-MAE) between the predicted and true cluster number. Other metrics, including median ARI and NMI values, are provided in App. D.1.

4.1 Generalization on synthetic datasets

We first evaluate TabClustPFN on 49 synthetic datasets, comprising 25 GMM-generated and 24 ZEUS-generated datasets. All datasets are held out from pretraining. Detailed statistics are provided

in Tab. 8 and Tab. 9 in App. C.1. Results are reported in Tab. 1, which shows the ranks of 18 methods that select K. Since the number of datasets is odd, the reported medians of rank-based metrics take integer values. Ranks for methods with known K are provided in App. D.2. Across all metrics and methods, TabClustPFN significantly outperforms other methods, achieving the SOTA performance.

A closer look at the model. To understand the representations learned by our model, we visualize them on representative synthetic datasets in Fig. 3, comparing raw features, pretrained TabICL

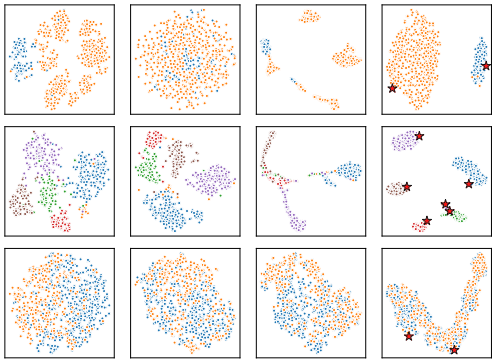

Figure 3: **Representation t-SNE**. Columns: raw features, TabICL encoder, ZEUS, TabClustPFN. Rows: increasing difficulty.

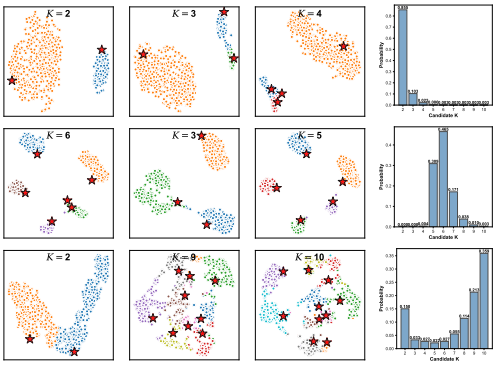

Figure 4: **PIN representations & CIN posteriors**. Columns show $R^{(L)}$, other K, and $\hat{p}(K|\mathbf{X})$.

embeddings, ZEUS representations, and `TabClustPFN`. Visualizations for the full synthetic sets are provided in App. D.3. Our method has strong performance on most of the synthetic datasets. The 1st and 2nd rows correspond to datasets where TabClustPFN outperforms ZEUS, while the last row shows a case where both methods fail. For our TabClustPFN, red stars denote cluster prototypes C^L and points correspond to final representations R^L after the last MLP layer and normalization. Points are colored by ground-truth labels. In the last row, the two clusters are highly overlapping, making the dataset intrinsically difficult regardless of the clustering algorithm. In the 1st row, t-SNE applied to raw features, TabICL embeddings, and ZEUS representations fails to reveal clear cluster structure. In contrast, TabClustPFN produces well-separated, blob-shaped groups, with prototypes located near cluster centers. This indicates that TabClustPFN reshapes the data into a geometry where cluster structure becomes explicit and aligned with prototype directions. In contrast, ZEUS representations often exhibit elongated, anisotropic cluster shapes. This can cause silhouette-based model selection to choose an incorrect number of clusters, as the metric implicitly favors compact, blob-like groups. We also observe that ZEUS performs well when the raw feature space already exhibits clear cluster structure (2nd row), but struggles when clusters are not well shaped in the input space (1st row).

We visualize the decoder representations $R^{(L)}$ produced by PIN under different candidate values of K, together with the posterior over K predicted by CIN (Fig. 4). For the easiest case (1st row), CIN outputs a sharp posterior concentrated on a single K, and the representation shows compact, well-separated clusters. As difficulty increases (3rd row), the posterior becomes more diffuse, reflecting greater uncertainty in K. We also observe that K values with higher $\hat{p}(K|\boldsymbol{X})$ yield more coherent and compact cluster geometry, while low-probability K leads to fragmented or less structured layouts. This alignment between representation quality and posterior mass indicates that CIN captures structural cues from PIN when estimating the cluster cardinality.

Calibration. We evaluate the calibration of the cardinality posterior $\hat{p}(K|\boldsymbol{X})$ predicted by CIN (see App. D.7). We do not evaluate $\hat{p}(\boldsymbol{Z}|K, \boldsymbol{X})$, as calibration of cluster assignments is not well-defined under label switching. Standard calibration methods require a fixed correspondence between predicted and ground-truth labels; while one could apply optimal matching (*e.g.,* Hungarian algorithm), this introduces a dataset-level dependency and does not yield a principled notion of per-sample correctness. Hence, calibration lacks a clean definition in the clustering setting. Tab. 2 shows that the empirical coverage consistently exceeds the nominal level across all tested values of α. Thus, the CIN posterior is not overconfident: prediction sets formed from $\hat{p}(K|\boldsymbol{X})$ reliably cover the true cluster cardinality. Moreover, the empirical coverage is close to the target at high confidence

Table 2: **Calibration of the CIN posterior**.

Nominal α	80%	85%	90%	95%	99%
Empirical coverage	0.875	0.907	0.937	0.968	0.993

levels. These results suggest that TabClustPFN provides reliable uncertainty estimates for cluster cardinality, which is important in the unknown-K setting where overconfident cardinality predictions can directly degrade downstream clustering quality.

Generalization for larger N & D. We further evaluate out-of-distribution scaling beyond the pretraining range ($N \leq 1000$, $D \leq 64$), comparing against ZEUS$^+$ and KMeans$^+$, the strongest PFN-style and classical baselines in our benchmark. TabClustPFN generalizes strongly to larger sample sizes, maintaining the best ARI and the lowest k-MAE up to $N = 10^4$ (Fig. 5).

For larger feature dimensions, direct extrapolation is more challenging, but PCA provides a simple remedy in the moderate-OOD regime, where TabClustPFN+PCA achieves comparable performance by mapping inputs back closer to the pretraining support (Fig. 6). At higher dimensions, the main degradation comes from less accurate cardinality estimation after PCA: the projection changes the global structure used by CIN, increasing k-MAE, and errors in the selected K directly affect unknown-K ARI. This suggests that the current gap is mainly due to limited exposure to larger-dimensional data, and can be further reduced with continued pretraining.

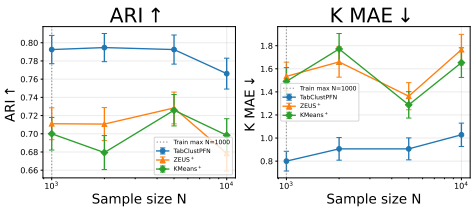

Figure 5: **OOD performance under larger N** (fixed $D = 50$).

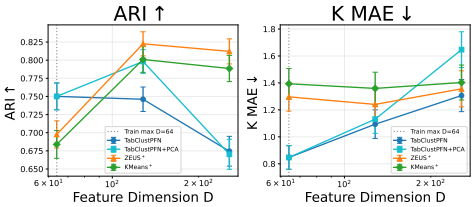

Figure 6: **OOD performance with larger D.**

Robustness to distribution shifts. We additionally evaluate the frozen checkpoint on cluster-size imbalance up to 100:1 and Student-t subpopulations. At $D = 2$, TabClustPFN maintains strong ARI under severe imbalance and remains competitive for degrees of freedom $\nu \geq 3$. At higher dimensions, median ARI across conditions exceeds 0.99 under both imbalance and heavy-tailed shifts, with detailed results provided in App. D.10.1.

4.2 Real-world clustering

We further evaluate TabClustPFN on a curated real-world benchmark to assess practical performance.

Benchmark. The benchmark contains datasets from multiple widely used sources, including OpenML-CC18 [4], TabArena [7], the AutoML benchmark suite [10], the ZEUS evaluation datasets, and a UCI clustering collection [24]. Duplicates across collections are removed, and synthetic datasets are excluded to ensure evaluation reflects generalization to truly unseen, real-world data rather than tasks similar to pretraining. To match our model regime, we retain datasets with at most 10 clusters, 64 features, and $1,000$ samples. This leads to 22 datasets from OpenML and 22 from UCI (App. C.1).

Results. The median ARI rank across 31 methods under both known-K and unknown-K settings is in Fig. 7 (left). As expected, providing the true number of clusters improves performance. TabClustPFN with known K achieves the best overall performance, while the variant inferring K via CIN ranks second. Notably, TabClustPFN with inferred K performs comparably to ZEUS followed by KMeans even when the latter is given the true K in its second stage, highlighting the effectiveness of our end-to-end inference. Fig. 7 (right) shows the k-MAE for methods that estimate K. TabClustPFN attains a k-MAE of about 1, lower than other PFN-based approaches, highlighting the effectiveness of CIN for cardinality estimation. We further compare with DP-means [16] and Dip-means [15], two additional methods without a fixed upper bound on K, and find

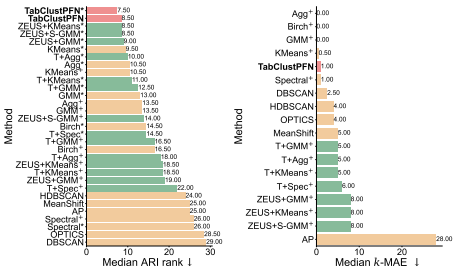

Figure 7: **Benchmark performance on real-world benchmark**. Median of ARI rank (left) and MAE in the estimated number of clusters (right); lower is better for both metrics. Red bar denotes our TabClustPFN, green bars denote other PFN-based methods, and yellow bars denote classical methods.

that TabClustPFN achieves higher median ARI and NMI on both benchmarks despite CIN's bounded support (App. D.5). More detailed comparisons are provided in App. D.8–D.9. Pairwise win rates show that `TabClustPFN` is robust across real-world datasets. *We assess statistical significance using paired t-tests and Wilcoxon signed-rank tests, which show that TabClustPFN significantly outperforms all baselines on synthetic data and all but Agg$^+$ and KMeans$^+$ on real-world data, where it still achieves higher ARI on most datasets.* We further report critical-difference diagrams based on average ARI ranks, which consistently rank `TabClustPFN` first on both benchmarks (App. D.6).

Real-world OOD evaluation. Applying the main benchmark's dataset inclusion criteria (App. C.1) within the OOD regime of $64 < D \leq 500$, $N < 3000$, and $2 \leq K \leq 10$ yields 6 additional real-world datasets. We evaluate the frozen checkpoint on all 6 datasets without retraining or fine-tuning. We compare against four representative baselines spanning centroid-based, hierarchical, density-based, and PFN-based clustering. TabClustPFN achieves the highest median ARI/NMI and lowest median cardinality error. Dataset identities, and details are provided in App. D.10.2.

4.3 Comparison with amortized clustering

In the main experiments, we compare primarily with classical clustering algorithms and PFN-based methods, but not with existing amortized clustering approaches such as DAC [18] and NCP [26]. The reason is twofold. First, these models are typically trained under a fixed data-generating family and often generalize poorly beyond that distribution. Second, scaling their training to the diversity and size of datasets considered in our broader benchmarks is computationally intensive. Nevertheless, to understand how our approach compares within the regime these methods are designed for, we follow their original evaluation protocol and conduct a controlled comparison under the same setting. Datasets are generated from a **2D GMM**-based process: each dataset $\mathcal{D}$ is formed by sampling the number of clusters K, mixture weights π, and component parameters $\{(\mu_k, \Sigma_k)\}_{k=1}^{K}$, and then drawing instances from the resulting GMM. Both training and test datasets are independent draws from this same distribution family (see App. C.4 for details).

Following DAC and NCP, we train all amortized baselines on $16{,}000$ synthetic training datasets with at most 10 clusters (each used once), and evaluate on 160 held-out datasets from the same GMM process. For our method, we report three models: (i) the TabClustPFN foundation model pretrained under our full prior, and two variants trained only on this GMM data: (ii) PFN (Random-init), trained from scratch without TabICL initialization, and (iii) PFN (TabICL-init), initialized with TabICL encoder weights. Results are summarized in Tab. 3.

Table 3: **Quantitative comparison of clustering performance and efficiency under the evaluation setting used by amortized clustering methods**. Results are reported in terms of median ARI, k-MAE, k-Acc, and clustering time. Bold values indicate the best result for each metric (↑ higher is better, ↓ lower is better).

Method	ARI ↑	k-MAE ↓	k-Acc ↑	Time (ms) ↓
DAC (Anc)	0.5665	10.0	0.00625	257.20
DAC (Min)	0.5209	5.0	0.0375	235.48
NCP	0.6000	3.0	0.08125	1058.04
PFN (Random-init)	0.4222	1.0	**0.4250**	**69.58**
PFN (TabICL-init)	0.6294	1.0	**0.4250**	71.79
TabClustPFN	**0.7893**	**1.0**	**0.4250**	70.93

TabClustPFN achieves the best overall performance, showing that the PFN paradigm yields a stronger amortized clustering model even in the narrow setting tailored to previous amortized methods. PFN (TabICL-init) also outperforms all amortized clustering baselines and improves over PFN (Random-init), indicating that stronger initialization facilitates optimization even when the data distribution is simple and homogeneous. With larger-scale pretraining, the performance gap of the randomly initialized model is expected to shrink. In addition, our method provides a $4\times$ speedup compared to the fastest baseline.

5 Discussion on design choices

Effectiveness of iterative cross attention. We evaluate whether the decoder architecture drives the performance gains of TabClustPFN by keeping the encoder and training setup fixed while comparing our iterative cross-attention decoder against two non-iterative baselines. The first baseline is a *naïve SA+MLP decoder*, which directly predicts cluster assignments from point embeddings: it applies self-attention over the N points to model point–point interactions, then uses a point-wise MLP head to output K-way assignment logits per point. Its complexity is $\mathcal{O}(N^2 + NK)$. This design contains no explicit cluster representations; clusters are implicitly represented in the MLP output layer. The second baseline is a *non-iterative Transformer decoder*.

This variant introduces *explicit cluster prototypes* but omits iterative prototype–data co-refinement: it first encodes the points with self-attention to obtain $R^{(L)}$, then updates the K prototypes using prototype self-attention (cluster–cluster interaction) and prototype–point cross-attention with keys/values from the fixed encoded points $R^{(L)}$ (cluster–point interaction). Its complexity remains $\mathcal{O}(N^2 + NK)$. In contrast, our decoder maintains K cluster prototypes and *iteratively refines* point–cluster assignments and prototypes: each layer alternates (i) prototype self-attention (cluster–cluster interaction) and (ii) bidirectional prototype–point cross-attention between N point embeddings and K prototypes (point–cluster interaction), while avoiding full point–point self-attention in the decoder. Each layer costs $\mathcal{O}(NK + K^2)$, less than both baselines for $K \ll N$.

Table 4: **PIN decoder architecture ablation**. Comparison of different model architectures on synthetic and real datasets using median ARI and NMI (higher is better). Bold and underlined values indicate the best and second-best performance in each column.

Method	synthetic		Real	
	ARI ↑	NMI ↑	ARI ↑	NMI ↑
Naïve (SA + MLP)	0.8137	0.7446	<u>0.3589</u>	0.3271
Non-iterative (Transformer)	<u>0.8229</u>	<u>0.7523</u>	0.3505	<u>0.3865</u>
Iterative (Ours)	**0.8308**	**0.8000**	**0.3853**	**0.4237**

Tab. 4 shows that the iterative decoder consistently achieves the best performance across both settings. Despite its higher computational cost, the non-iterative Transformer underperforms our design. This suggests that explicit pairwise point modeling alone is insufficient; instead, *iterative refinement of cluster prototypes* is key. While cluster queries already improve over the naïve SA+MLP decoder, the full iterative mechanism achieves the best clustering quality and efficiency.

PIN loss ablation. To assess whether SoftARI is an appropriate objective, we compare it with several natural alternatives while keeping keeping all other components fixed. We first consider SoftNMI, based on normalized mutual information (NMI), a clustering metric that is invariant to label permutations. We also include two matching-based baselines that explicitly address the label switching issue. These methods resolve label switching by aligning predicted clusters with ground-truth labels via either minimum-weight matching [5] or a differentiable Sinkhorn relaxation [32], and then apply a cross-entropy (CE) loss or a soft accuracy loss. Soft versions of NMI and accuracy are adopted to avoid discrete operations (e.g., hard labeling or hard matching). These baselines are denoted as Matching+SoftAcc and Matching+CE, respectively. Details of all loss formulations and an additional ablation study on CIN loss are provided in App. E. Tab. 5 reports the median ARI of `TabClustPFN` pretrained with different PIN losses on both synthetic and real datasets. SoftARI yields the best overall performance. While Matching+CE achieves comparable accuracy, its pretraining cost is substantially higher due to the additional computational overhead introduced by the matching step. In the last column, we report the total time for one forward and backward pass with batch size 8 and 1000 samples per dataset. The per-iteration time of Matching+CE is approximately $2\times$ that of our approach, resulting in a doubled optimization time, which is significant given $10{,}000$ steps.

Table 5: **PIN loss ablation**. Median ARI and NMI on synthetic and real datasets.

Method	Synthetic		Real		Time(ms) ↓
	ARI ↑	NMI ↑	ARI ↑	NMI ↑	
Matching+SoftAcc	0.8021	0.7364	0.3814	0.4071	70.0346
Matching+CE	**0.8627**	<u>0.7637</u>	0.3507	0.4070	6.2625
SoftNMI	0.7637	0.6929	**0.3915**	**0.4250**	3.3658
SoftARI (Ours)	<u>0.8308</u>	**0.8000**	<u>0.3853</u>	<u>0.4237</u>	**2.9695**

Decoupled optimization ablation. We compare *additive joint training* of PIN and CIN with our *decoupled strategy*, where CIN operates on detached PIN outputs. As shown in Tab. 6, decoupling consistently improves performance on both synthetic and real datasets. We attribute this to reduced optimization interference: partition inference and cardinality estimation operate at different structural levels, and shared gradients can bias representations toward predicting K at the expense of fine-grained partition structure. Decoupling stabilizes cluster representations, improving both clustering quality and estimation of K.

Table 6: **Decoupled loss ablation**. Median ARI and NMI on synthetic and real datasets.

Method	Synthetic		Real	
	ARI ↑	NMI ↑	ARI ↑	NMI ↑
Additive	<u>0.8103</u>	<u>0.7227</u>	<u>0.3544</u>	<u>0.3716</u>
Decoupled (Ours)	**0.8308**	**0.8000**	**0.3853**	**0.4237**

Different prior data mechanisms. To evaluate the impact of prior data diversity, we consider three prior configurations: training with GMM prior data only, ZEUS prior data only, or a combination of both. We then assess performance on both synthetic and real datasets. For the synthetic benchmarks, we further provide a fine-grained analysis by grouping datasets

Table 7: **Prior data types ablation**. Median ARI on synthetic and real datasets.

Prior data	Synthetic ↑			Real ↑
	GMM	ZEUS	GMM+ZEUS	
GMM	**0.8236**	0.5434	<u>0.7428</u>	0.2270
ZEUS	0.6758	<u>0.8047</u>	0.7405	<u>0.3414</u>
GMM+ZEUS (Ours)	<u>0.8230</u>	**0.8334**	**0.8308**	**0.3853**

according to their data-generation mechanisms. The results are reported in Tab. 7. Increasing the diversity of prior data consistently improves overall performance across both synthetic and real datasets. While the ZEUS-only prior performs best on datasets generated from the same family, its generalization to other dataset types is limited. These findings are consistent with [33], which show that greater prior diversity can enhance the performance of PFN models. We report negative results from alternative prior constructions in App. B.1.3.

6 Limitations & conclusions

We introduced `TabClustPFN`, an extensible PFN-based framework for tabular clustering that jointly infers cluster assignments and cardinality in a single forward pass without dataset-specific tuning. Unlike prior PFN-based clustering methods that rely on known-cardinality assumptions, low-dimensional synthetic priors, or label-ordering conventions, `TabClustPFN` combines a hybrid tabular prior, permutation-invariant SoftARI training, and a decoupled PIN–CIN architecture to support high-dimensional heterogeneous tabular data. Across 44 real-world datasets, `TabClustPFN` achieves SOTA performance over classical, deep, and amortized baselines while maintaining fast inference.

Future work. Due to computational constraints, our experiments focus on datasets of moderate size and dimensionality. The OOD evaluations provide evidence of generalization beyond the pretraining regime, with degradation under some shifts. Moderately larger datasets (N, D) can use the same architecture with broader pretraining and more compute. Increasing $K_{\max}$ requires enlarging the PIN prototype bank and CIN input/output layers, then retraining on higher-cardinality priors. Improving robustness to noisy or missing entries through corruption-aware priors is another important direction for future work.

Acknowledgments and Disclosure of Funding

Tianqi Zhao and Qiong Zhang are supported by the National Key R&D Program of China Grant 2024YFA1015800. Yan Shuo Tan is supported by NUS Start-up Grant A-8000448-00-00 and MOE AcRF Tier 1 Grants A-8002498-00-00 and A-8004458-00-00.

This work was conducted while Qiong Zhang was at the Institute of Statistics and Big Data, Renmin University of China. Qiong Zhang is now affiliated with Mohamed bin Zayed University of Artificial Intelligence.

References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[2] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2019.

[3] P. Bhaskaran and T. Viering. Transformers can do Bayesian clustering. *arXiv preprint arXiv:2510.24318*, 2025.

[4] B. Bischl, G. Casalicchio, M. Feurer, P. Gijsbers, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. OpenML benchmarking suites. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[5] D. F. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[7] N. Erickson, L. Purucker, A. Tschalzev, D. Holzmüller, P. M. Desai, D. Salinas, and F. Hutter. Tabarena: A living benchmark for machine learning on tabular data. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226–231, 1996.

[9] A. Flynt, N. Dean, and R. Nugent. sARI: a soft agreement measure for class partitions incorporating assignment probabilities. *Advances in Data Analysis and Classification*, 13(1):303–323, 2019.

[10] P. Gijsbers, M. L. P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

[11] L. Grinsztajn, K. Flöge, O. Key, F. Birkel, P. Jund, B. Roof, B. Jäger, D. Safaric, S. Alessi, A. Hayler, et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

[12] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[13] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[14] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[15] A. Kalogeratos and A. Likas. Dip-means: an incremental clustering method for estimating the number of clusters. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[16] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *International Conference on Machine Learning*, 2012.

[17] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[18] J. Lee, Y. Lee, and Y. W. Teh. Deep amortized clustering. *arXiv preprint arXiv:1909.13433*, 2019.

[19] 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), 2025.

[20] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[21] J. Ma, V. Thomas, R. Hosseinzadeh, A. Labach, H. Kamkari, J. C. Cresswell, K. Golestan, G. Yu, A. L. Caterini, and M. Volkovs. TabDPT: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[22] P. Marszałek, T. Kuśmierczyk, W. Wydmański, J. Tabor, and M. Śmieja. ZEUS: Zero-shot embeddings for unsupervised separation of tabular data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[23] V. Melnykov, W.-C. Chen, and R. Maitra. MixSim: An R package for simulating data to study performance of clustering algorithms. *Journal of Statistical Software*, 51(12):1–25, 2012.

[24] milaan9. Clustering-datasets. `https://github.com/milaan9/Clustering-Datasets`, 2019. GitHub repository, accessed January 2026.

[25] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, page 849–856, 2001.

[26] A. Pakman, Y. Wang, C. Mitelut, J. Lee, and L. Paninski. Neural clustering processes. In *International Conference on Machine Learning*, pages 7455–7465. PMLR, 2020.

[27] J. Pearl. *Causality*. Cambridge university press, 2009.

[28] J. Qu, D. Holzmüller, G. Varoquaux, and M. L. Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025.

[29] J. QU, D. Holzmüller, G. Varoquaux, and M. L. Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Forty-third International Conference on Machine Learning*, 2026.

[30] H. T. Rauf, A. Freitas, and N. W. Paton. TableDC: Deep clustering for tabular data. *Proceedings of the ACM on Management of Data*, 3(3):1–28, 2025.

[31] Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, P. S. Yu, and L. He. Deep clustering: A comprehensive survey. *IEEE transactions on neural networks and learning systems*, 36(4):5858–5878, 2025.

[32] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, May 1967.

[33] X. Zhang, D. C. Maddix, J. Yin, N. Erickson, A. F. Ansari, B. Han, S. Zhang, L. Akoglu, C. Faloutsos, M. W. Mahoney, C. Hu, H. Rangwala, G. Karypis, and B. Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Related work

Classical probabilistic and distance-based clustering methods. Classical clustering methods remain widely used for tabular data due to their simplicity and computational efficiency. However, they typically operate directly on the input features and rely on rigid geometric or probabilistic assumptions to separate data. For example, k-*means* [20] favors convex centroid-based partitions, Gaussian mixture models (GMMs) assume Gaussian component densities, *DBSCAN* [8] relies on a global density scale, and hierarchical clustering [14] is highly sensitive to the choice of distance metric and linkage criterion. Such inductive biases are effective in simple settings, but often fail to capture the heterogeneous structures commonly observed in real-world tabular data.

Manifold and representation learning methods. To overcome the rigidity of classical methods, manifold and representation learning approaches seek transformations under which cluster structures become more separable. *Spectral clustering* [25] captures non-convex structures through graph Laplacian eigenspace embeddings, while recent *deep clustering* methods [30, 31] learn nonlinear representations with neural networks. Although these methods are more expressive, they still formulate clustering as a *dataset-specific* optimization problem. Spectral clustering requires eigendecomposition with cubic complexity in the number of samples, and deep clustering methods require training a separate model for each dataset. As a result, they are often too costly for exploratory analysis and do not provide a transferable clustering inductive bias across datasets.

Amortized clustering methods. A more recent paradigm replaces dataset-specific optimization with amortized neural inference. Amortized clustering methods [17, 18, 26] train neural networks to infer cluster assignments in a zero-shot or iterative manner, thereby reducing the need for per-dataset search or model fitting. However, existing approaches are typically trained to approximate simple generative priors such as GMMs. Consequently, while they improve inference efficiency, their clustering behavior remains constrained by the same parametric assumptions underlying the priors they emulate.

B Pretraining details

This section provides additional details on the pretraining setup. The prior data used for pretraining (Sec. B.1) are presented first, followed by network architecture details (Sec. B.2).

B.1 Prior data generation

We first sample basic dataset-level properties. The number of observations N is drawn uniformly from integers between 500 and 1000. The feature dimension D is sampled uniformly from integers between 2 and 64. The number of clusters K is drawn from $\{2, \dots, K_{\max}\}$ using a non-uniform prior: $\mathbb{P}(K = 2) = 0.3$, while with probability 0.7, K is sampled uniformly from $\{3, \dots, 10\}$. This non-uniform prior over the number of clusters follows the setting of [28]. It places greater probability mass on simpler cases, which can facilitate training by biasing the model toward learning from less complex clustering structures.

We then select the data-generating mechanism by sampling a Bernoulli variable, using the GMM prior with probability 0.4 and the ZEUS prior with probability 0.6. The selected mechanism is then used to generate a synthetic clustering dataset in $\mathbb{R}^D$ with N observations, along with latent component indices serving as ground-truth cluster labels. We subsequently apply the same preprocessing steps to every generated dataset, including feature z-norm, feature permutation, before assembling datasets into training mini-batches. We next describe the two priors in detail.

B.1.1 GMM prior

Clustering difficulty in mixture models is largely governed by the degree of overlap between components: well-separated components yield near-deterministic assignments, whereas heavily overlapping components make cluster identities intrinsically ambiguous. To systematically control task difficulty during pretraining, we therefore generate datasets from Gaussian mixture models (GMMs) with an explicit constraint on component overlap.

Overlap measure. For a K-component Gaussian mixture with parameters $\{(\pi_k, \mu_k, \Sigma_k)\}_{k=1}^{K}$, we quantify pairwise overlap using Bayes misclassification probabilities under the model-based decision

rule. Let $\phi(x|\mu, \Sigma)$ denote the Gaussian density and π_k the mixing weight. The one-sided overlap from component i to component j is defined as

$$o_{j|i} = \mathbb{P}\left(\pi_i \phi(X|\mu_i, \Sigma_i) < \pi_j \phi(X|\mu_j, \Sigma_j) | X \sim \mathcal{N}(\mu_i, \Sigma_i)\right),$$

which is the probability that a sample generated from component i would be assigned to component j by the Bayes classifier. The symmetric pairwise overlap is

$$\omega_{ij} = o_{j|i} + o_{i|j},$$

and the overall dataset difficulty is characterized by the maximum pairwise overlap

$$\Omega_{\max} = \max_{i \neq j \in [K]} \omega_{ij}.$$

Small $\Omega_{\max}$ corresponds to well-separated clusters, while large values indicate intrinsically ambiguous clustering structure.

Controlled GMM generation. We generate GMM parameters using the MIXSIM package [23] in R, which directly constructs mixtures satisfying a prescribed upper bound on $\Omega_{\max}$. This enables sampling datasets with controlled and diverse levels of clustering difficulty. We specify the configuration of MixSim below.

For each synthetic dataset, we sample mixture parameters as follows:

- **Mixture weights.** We sample $\pi \sim \text{Dirichlet}(\alpha \mathbf{1}_K)$ with concentration $\alpha = 2$, encouraging moderately balanced clusters. To avoid degenerate components, we enforce a minimum proportion constraint using $\texttt{PiLow} = 10^{-1}$.
- **Target overlap (difficulty control).** We first sample a target maximum pairwise overlap

  $$\Omega_{\max} \sim \text{Uniform}\left[0.01,\ \min\left(0.8,\ \frac{1.5}{D^{0.82}}\right)\right].$$

  The dimension-dependent upper bound ensures feasibility of the overlap constraint in higher dimensions, where random components tend to become more separable. MIXSIM then constructs (μ_k, Σ_k) such that the resulting mixture satisfies this target $\Omega_{\max}$.
- **Component shape diversity.** To promote geometric diversity, we randomize covariance structures. For each dataset, we independently sample whether components are spherical (`sph = TRUE`) or ellipsoidal (`sph = FALSE`), and whether covariances are shared across clusters (`hom = TRUE/FALSE`). When ellipsoidal covariances are used, we allow strong anisotropy via a maximum eccentricity constraint ($\texttt{ecc} = 0.9$), producing elongated clusters.
- **Component location range.** Component means are constrained to a bounded hypercube by sampling each μ_k uniformly from $[-1, 1]^D$. This controls the global scale of cluster locations and prevents extreme separation driven purely by mean magnitude.

Data sampling. Given the generated parameters $\{(\pi_k, \mu_k, \Sigma_k)\}_{k=1}^K$, we first draw N latent component labels $Z_i \overset{\text{IID}}{\sim} \text{Categorical}(\pi)$, which serve as ground-truth cluster assignments. Then, conditioning on each Z_i, we draw samples from $\mathcal{N}(\mu_{Z_i}, \Sigma_{Z_i})$.

Visualization. Overall, these controls provide a scalable way to generate GMM datasets with explicitly tunable inter-cluster ambiguity while retaining substantial diversity in cluster geometry and cluster-size imbalance. Fig. 8 shows representative 2D mixtures generated under different $\Omega_{\max}$ values.

B.1.2 ZEUS prior

While the GMM prior controls clustering difficulty through Gaussian overlap, the ZEUS prior generates more geometrically and statistically heterogeneous tabular data by combining latent mixture structure with nonlinear transformations and mixed feature types.

Given the total feature dimensionality D, we randomly partition features into a continuous subset and a categorical subset. We first sample the number of continuous features $D_{\text{cont}} \geq 2$, and treat the remaining $D - D_{\text{cont}}$ dimensions as categorical. Each categorical feature is label-encoded (rather than one-hot), and its cardinality is sampled from a predefined range.

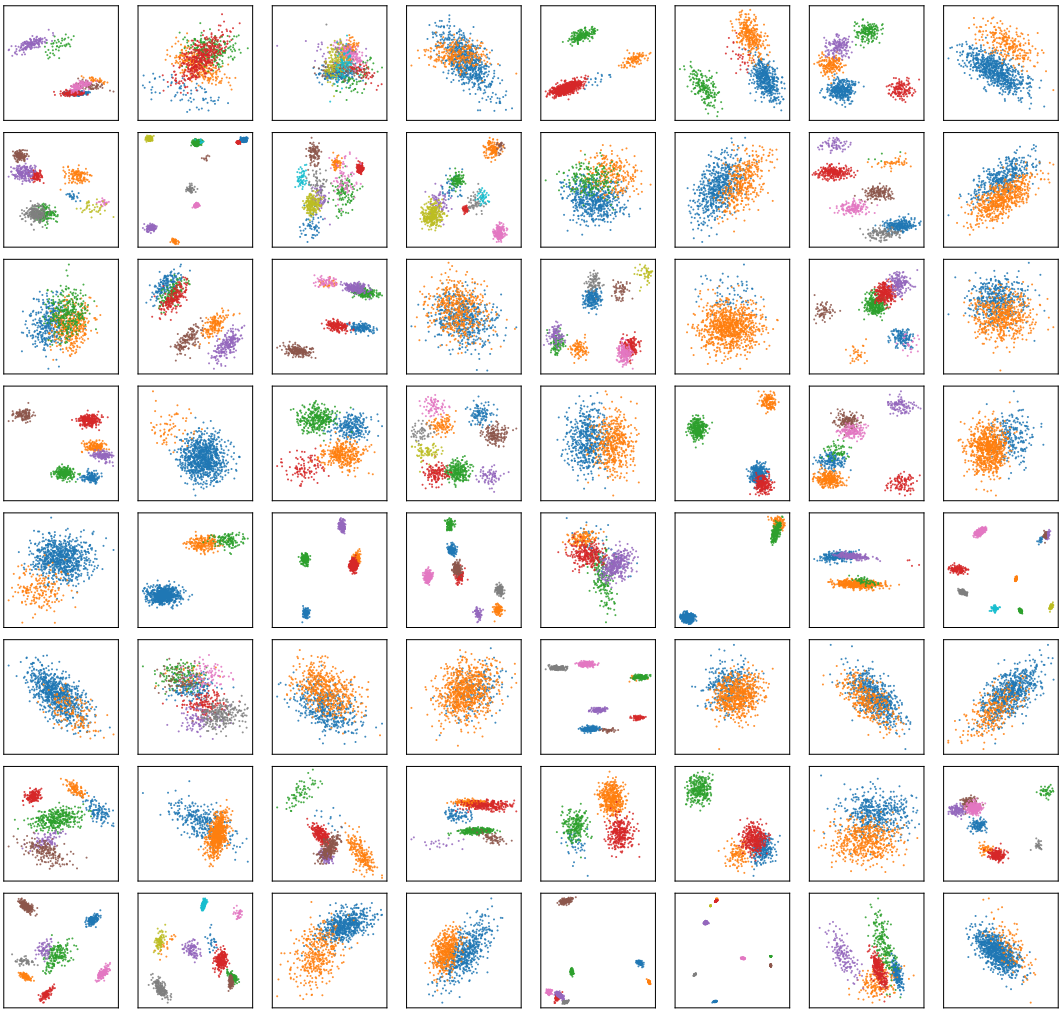

Figure 8: **Visualization of randomly generated 2D datasets from the GMM prior**.

Continuous features. For the continuous feature subsets, we sample mixture weights $\pi \sim \text{Dirichlet}(\alpha \mathbf{1}_K)$ with $\alpha = 2$ and latent component labels $Z_i \sim \text{Categorical}(\pi)$, which serve as ground-truth cluster assignments. Unlike ZEUS, which uses a smaller α to induce more imbalanced mixtures, we set $\alpha = 2$ to avoid degenerate components and overly skewed cluster sizes, yielding more stable supervision and a broader range of moderately imbalanced clustering instances. Conditional on Z_i, base continuous observations are drawn from cluster-specific distributions as described in ZEUS [22]. For completeness, we summarize the relevant settings below.

To increase geometric complexity beyond a simple mixture structure, ZEUS introduces nonlinear feature interactions through an invertible transformation. With probability 0.5, the identity mapping is used, and with probability 0.5, a randomly sampled invertible iResNet bijection with Lipschitz constant constrained to $[0.1, 0.9]$ is applied. This transformation yields nonlinearly distorted cluster shapes while preserving cluster identity due to invertibility. The number of invertible blocks follows a log-scaled truncated normal meta-distribution. This design choice follows prior tabular data generation work (e.g., TabICL) rather than being specific to ZEUS, as it can increase the diversity of the data generation process. Concretely,

$$\log \mu \sim \text{Uniform}(\log 3, \log 8), \qquad \log \sigma \sim \text{Uniform}(\log 0.01, \log 1.0),$$

$\tilde{b} \sim \mathcal{N}(\mu, \sigma^2)$ is sampled with truncation to $[0, \infty)$, and the number of blocks is set as $b = 3 + \text{round}(\tilde{b})$. This results in variable transformation depth across datasets.

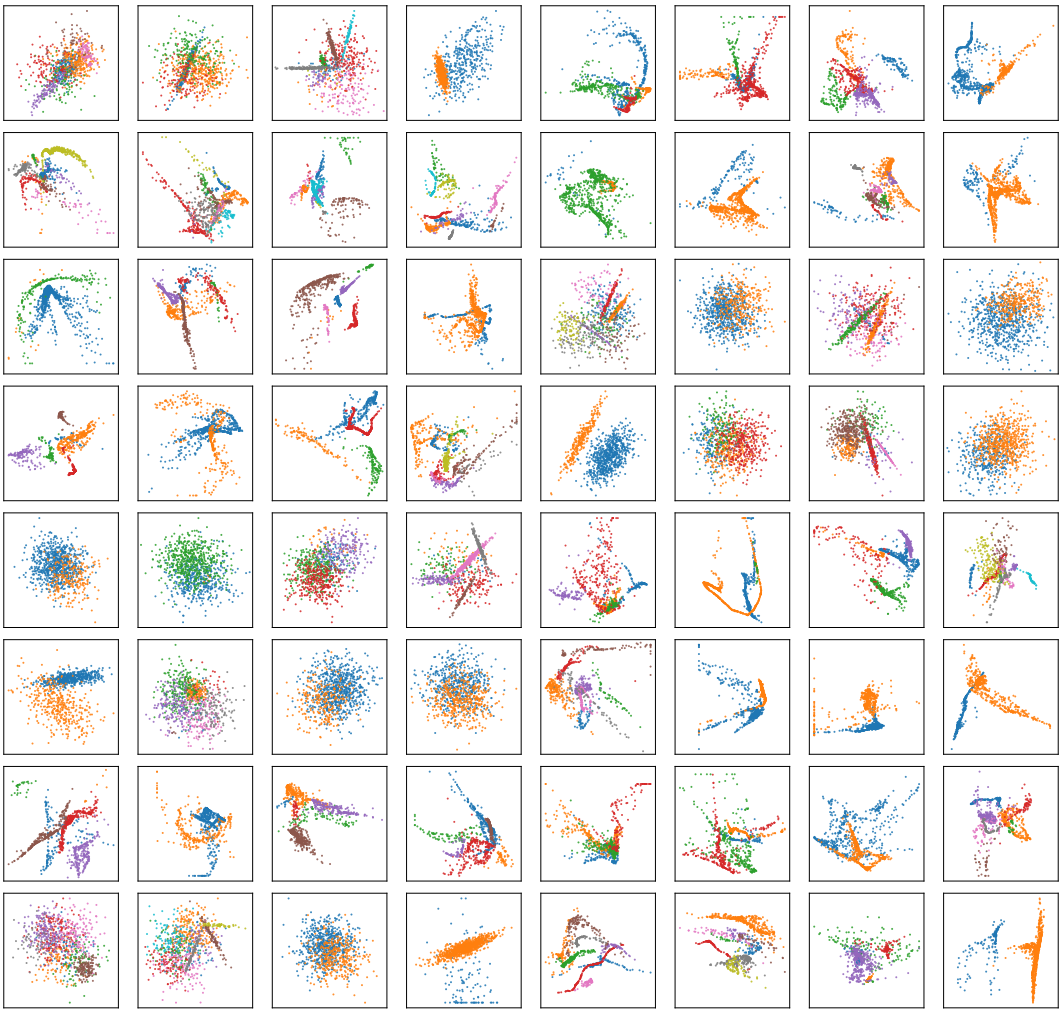

Figure 9: **Visualization of randomly generated 2D datasets from the ZEUS prior**.

Categorical features. Following ZEUS, categorical variables are generated conditional on the latent cluster labels. For each categorical feature j, we first sample its cardinality

$$C_j \in \{2, \ldots, C_{\max}\}, \qquad C_{\max} = 5,$$

then draw cluster-specific categorical distributions (e.g., via Dirichlet priors) so that different clusters exhibit distinct category preferences. Each observation samples its categorical value from the distribution associated with its latent component z_i, providing discrete cluster cues complementary to the continuous geometry.

Data assembly. Finally, we concatenate the transformed continuous features and categorical features to obtain a D-dimensional tabular dataset. This construction produces clusters that may be nonlinearly separable in the observed space and only partially identifiable from any single feature type, encouraging the model to learn flexible multimodal structure.

Visualization. Figure 9 shows representative 2D datasets generated from the ZEUS prior, illustrating nonlinear cluster geometry and mixed-type feature structure.

B.1.3 Alternative prior constructions and negative results

We explored several alternative prior constructions before selecting the GMM and ZEUS priors used in `TabClustPFN`. We summarize these exploratory results to document the choices underlying the final prior.

SCM-derived partitions. We reused supervised MLP-based and tree-based structural causal model (SCM) generators and constructed cluster labels by discretizing their outputs. These constructions performed poorly in our experiments. In the configurations tested, the discretized targets did not reliably correspond to recoverable cluster structure in the observed features. This observation motivated a generative construction with explicit latent cluster assignments and cluster-specific distributions, as in Eq. (1).

Unconstrained MLP transformations. We also sampled latent Gaussian components and transformed their observations using unconstrained random MLPs while retaining the component labels. This construction likewise performed poorly. A non-injective transformation can map distinct latent components to overlapping or indistinguishable regions, making the retained labels difficult to recover from the observed data. This provides a possible explanation for the observed behavior and motivated the use of invertible transformations to preserve component information while introducing nonlinear geometry.

Other transformation families. We further explored affine-coupling normalizing flows and cluster-aware transformations. These alternatives trained stably but did not improve downstream clustering performance over the iResNet-based ZEUS prior in the tested configurations. One possible explanation is that they vary the transformation family without providing sufficiently complementary training distributions. By contrast, the prior ablation in Tab. 7 shows that combining the controlled-overlap GMM prior with the ZEUS prior improves aggregate performance over either prior alone.

These observations are specific to the constructions and configurations tested; they do not establish that the alternative prior families are ineffective in general. They suggest that recoverable cluster structure and complementary data distributions are useful considerations when designing clustering priors.

B.2 Network architecture details

B.2.1 Partition Inference Network (PIN)

Given a tabular dataset $X \in \mathbb{R}^{N \times D}$ and a candidate cluster cardinality $K \in \{2, \dots, K_{\max}\}$, the PIN outputs a soft partition

$$\text{PIN}_\theta : (X, K) \mapsto P \in [0,1]^{N \times K}, \quad P_{ik} = \hat{p}(Z_i = k | X, K),$$

where Z_i denotes the latent cluster label of the i-th observation.

Encoder. The encoder of our PIN follows the TabICL design and consists of (i) a column-wise embedding module and (ii) a row-wise interaction module. We refer to TabICL for architectural details of these two components. Formally, given a dataset, the PIN encoder produces row representations

$$R^{(0)} = \text{Enc}(X) = \text{RowInteract}(\text{ColEmbed}(X)) \in \mathbb{R}^{N \times d},$$

where d is the representation dimension produced by the encoder. In our implementation, we use $d = 512$ following TabICL.

Decoder with learnable prototypes. For each candidate K, PIN instantiates K learnable cluster prototypes by selecting the first K rows from a global prototype matrix

$$C^{(0)} = \text{Select}_K(\mathbf{C}) \in \mathbb{R}^{K \times d}, \quad \mathbf{C} \in \mathbb{R}^{K_{\max} \times d}.$$

The decoder then applies L stacked *iterative cross-attention blocks* to co-refine the data representations and prototypes. As discussed in Sec. 3.2, all attention layers omit positional encodings to preserve permutation equivariance. We set $L = 6$ in all experiments, unless otherwise stated. At layer $\ell \in \{0, \dots, L-1\}$, the update consists of two stages:

1. **Prototype update (prototype $\rightarrow$ data cross-attention).** We first update the prototypes via (i) self-attention among prototypes and (ii) cross-attention where prototypes attend to data:

$$C^{(\ell+\frac{1}{2})} = \text{SA}\Big(C^{(\ell)}\Big), \quad C^{(\ell+1)} = \text{CA}_{C \leftarrow R}\Big(C^{(\ell+\frac{1}{2})}, R^{(\ell)}\Big).$$

   We instantiate both $\text{SA}(\cdot)$ and $\text{CA}(\cdot,\cdot)$ using the same Transformer-style attention block with multi-head attention (MHA), pre-norm layer normalization, residual connections, and a position-wise

feed-forward network (MLP). As discussed in Sec. 3.2, we omit positional encodings to preserve permutation equivariance. For completeness, we detail the exact attention-block formulation shared by SA and CA below.

Definition of SA/CA blocks. We adopt a pre-norm variant of the Multihead Attention Block (MAB) used in Set Transformer [17], which we denote by $\mathrm{MAB}_{\text{pre}}(\cdot\,;\cdot)$. Let H be the number of attention heads and $d_h = d/H$ the head dimension; we use $H = 4$ in all experiments unless otherwise stated. Given queries $Q \in \mathbb{R}^{n_q \times d}$, keys $K \in \mathbb{R}^{n_k \times d}$, and values $V \in \mathbb{R}^{n_k \times d}$, multi-head attention (MHA) is

$$\mathrm{MHA}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_H)\, W^O,$$

where

$$\mathrm{head}_h = \mathrm{softmax}\left(\frac{QW_h^Q (KW_h^K)^\top}{\sqrt{d_h}}\right) VW_h^V,$$

$\{W_h^Q, W_h^K, W_h^V\}_{h=1}^H$ and W^O are learnable projections. An attention sublayer with pre-norm and residual is defined as

$$\mathrm{Attn}(X; Y) = X + \mathrm{MHA}(\mathrm{LN}(X), \mathrm{LN}(Y), \mathrm{LN}(Y)),$$

and the full pre-norm MAB applies a position-wise feed-forward network (MLP/FFN) with pre-norm and residual:

$$\mathrm{MAB}_{\text{pre}}(X; Y) = \mathrm{Attn}(X; Y) + \mathrm{MLP}(\mathrm{LN}(\mathrm{Attn}(X; Y))), \tag{4}$$

where MLP is a two-layer position-wise FFN with hidden width $2d$ (i.e., $d \to 2d \to d$).

Self- and cross-attention operators. Using $\mathrm{MAB}_{\text{pre}}$, we define self-attention on a set as

$$\mathrm{SA}(X) := \mathrm{MAB}_{\text{pre}}(X; X), \tag{5}$$

i.e., queries/keys/values are all drawn from X (capturing within-set interactions). Cross-attention from a query set X to a context set Y is defined as

$$\mathrm{CA}_{X \leftarrow Y}(X, Y) := \mathrm{MAB}_{\text{pre}}(X; Y), \tag{6}$$

i.e., queries come from X while keys/values come from Y (aggregating information from Y into X).

In particular, prototype self-attention and prototype-to-data cross-attention are

$$\mathrm{SA}(C) \in \mathbb{R}^{K \times d}, \qquad \mathrm{CA}_{C \leftarrow R}(C, R) \in \mathbb{R}^{K \times d},$$

where $\mathrm{SA}(C)$ captures inter-cluster interactions and $\mathrm{CA}_{C \leftarrow R}(C, R)$ aggregates dataset evidence into prototype summaries.

2. **Representation update (data → prototype cross-attention).** We then update data representations by attending to the updated prototypes:

$$R^{(\ell+1)} = \mathrm{CA}_{R \leftarrow C}\left(R^{(\ell)}, C^{(\ell+1)}\right),$$

where $\mathrm{CA}_{R \leftarrow C}(R, C) := \mathrm{Block}(R; C)$, i.e., $Q = R$ and $K = V = C$, injecting cluster-level structure back into point representations.

Cosine-similarity logits. After the final decoder layer, we compute logits via normalized cosine similarity between row representations and prototypes. Let

$$\bar{r}_i = \frac{g\left(R_{i:}^{(L)}\right)}{\left\| g\left(R_{i:}^{(L)}\right)\right\|_2}, \quad \bar{c}_j = \frac{g\left(C_{j:}^{(L)}\right)}{\left\| g\left(C_{j:}^{(L)}\right)\right\|_2},$$

where $A_{i:}$ denotes the ith row of matrix A and $g(\cdot)$ is a shared MLP projection applied to both rows and prototypes. We then compute the logits as

$$\ell_{ij} = \tau \bar{r}_i^\top \bar{c}_j, \quad \tau > 0,$$

where τ is a learnable temperature. Finally, the soft partition is obtained by a row-wise softmax:

$$P_{i:} = \mathrm{softmax}(\ell_{i:}) \in [0, 1]^K.$$

PIN is trained with the permutation-invariant SoftARI loss on P.

B.2.2 Cardinality Inference Network (CIN)

To infer the cluster numbers, we run PIN for each $K \in \{2, \dots, K_{\max}\}$ to obtain $\{P^{(K)}\}$. For each K, we compute the normalized Gram matrix

$$G^{(K)} = \frac{1}{N}(P^{(K)})^\top P^{(K)} \in \mathbb{R}^{K \times K},$$

which summarizes pairwise cluster co-assignment statistics. We then form a permutation-invariant fingerprint by extracting the diagonal entries and the strictly upper-triangular off-diagonal entries, sorting each in descending order, and concatenating them:

$$g^{(K)} = \Big[\text{sort}(\text{diag}(G^{(K)}))\,;\, \text{sort}(\text{triu}(G^{(K)}, 1))\Big] \in \mathbb{R}^{K(K+1)/2}.$$

Concatenating $\{g^{(K)}\}_{K=2}^{K_{\max}}$ yields a fixed-length vector, which is passed through an MLP to output logits over K:

$$\text{CIN}_\phi(X) = \text{MLP}\left(\text{concat}(g^{(2)}, \dots, g^{(K_{\max})})\right) \in \mathbb{R}^{K_{\max}-1}.$$

In our implementation, CIN is a 3-layer MLP with hidden width 256, which is trained with cross-entropy on the ground-truth K^*. Following our decoupled optimization design, gradients from CIN do not backpropagate into PIN.

C Experimental details

This section provides additional details on the evaluation setup. Statistics of the synthetic and real-world benchmarks (Sec. C.1) are presented first, followed by descriptions of the baseline methods (Sec. C.2), data preprocessing details (Sec. C.3), and the experiment setting for amortized clustering (Sec. C.4).

C.1 Statistics of synthetic and real-world benchmarks

Synthetic evaluation datasets. To assess the generalization ability of `TabClustPFN` on unseen data drawn from the same distribution families as the pretraining data, we construct a held-out synthetic benchmark, constructed by setting a different random seed. Following the two classes of priors introduced in Sec. 3.1, we refer to them as the *GMM prior* and the *ZEUS prior*. The benchmark contains 25 synthetic datasets generated from the GMM prior and 24 synthetic datasets generated from the ZEUS prior. Detailed statistics of these evaluation datasets are reported in Tab. 8 (GMM prior) and Tab. 9 (ZEUS prior). In both tables, the `ID` column provides a reference index for each synthetic dataset.

Real-world benchmark. We construct a real-world tabular clustering benchmark by aggregating datasets from several widely used sources: OpenML-CC18 [4], TabArena [7], the AutoML benchmark suite [10], the ZEUS evaluation dataset group, and a curated UCI clustering collection [24]. Tab. 10 and Tab. 11 report dataset-level statistics and identifiers. The OpenML table lists datasets by OpenML ID, while the UCI table follows the original dataset names. For each dataset, we report the number of instances, the counts of numerical and categorical features, the total feature dimensionality (numerical + categorical), and the number of clusters.

To ensure compatibility with the modeling assumptions and computational regime of `TabClustPFN`, we first filter all candidate datasets by feature dimensionality and cluster cardinality, retaining only those with $D \leq 64$ and $K \leq 10$. We then control dataset scale through source-dependent handling of the number of instances N. For large benchmark suites with many candidate datasets (e.g., OpenML-CC18, TabArena, and the AutoML benchmark), we further restrict the pool by keeping only datasets whose original size satisfies $N \leq 1000$. This prevents these large collections from dominating the benchmark. In contrast, the curated UCI collection and the ZEUS dataset group (a subset of OpenML) contain a more limited number of eligible datasets after the D and K filtering step. For these sources, we therefore do not exclude datasets based on size. Instead, when $N > 1000$, we uniformly subsample 1000 instances to match the evaluation scale. The subsampling is performed with a fixed random seed for reproducibility. We verify that the subsampled set preserves the original

Table 8: **Statistics of synthetic datasets for evaluation generated with the *GMM* prior**.

ID	# Instances	# Numerical features	# Categorical features	Dimension	# Classes
0	511	55	0	55	8
1	741	33	0	33	2
2	827	8	0	8	7
3	993	47	0	47	10
4	740	23	0	23	6
5	675	39	0	39	5
6	718	10	0	10	7
7	794	44	0	44	2
8	724	16	0	16	2
9	984	55	0	55	4
10	588	4	0	4	2
11	595	20	0	20	5
12	850	16	0	16	5
13	700	31	0	31	7
14	869	7	0	7	3
15	722	8	0	8	2
16	693	55	0	55	4
17	989	19	0	19	4
18	892	35	0	35	2
19	633	5	0	5	2
20	825	46	0	46	7
21	691	17	0	17	2
22	853	58	0	58	7
23	707	43	0	43	9
24	753	56	0	56	6

Table 9: **Statistics of synthetic datasets for evaluation generated with the *ZEUS* prior**.

ID	# Instances	# Numerical features	# Categorical features	Dimension	# Classes
0	649	3	4	7	2
1	910	18	14	32	7
2	948	6	43	49	6
3	905	13	17	30	2
4	863	14	1	15	2
5	898	6	0	6	2
6	589	11	9	20	10
7	568	12	9	21	10
8	523	38	10	48	10
9	785	2	27	29	2
10	658	2	17	19	6
11	628	11	41	52	5
12	862	21	3	24	2
13	632	35	17	52	2
14	906	5	57	62	2
15	688	4	25	29	10
16	757	10	17	27	8
17	705	19	31	50	2
18	590	14	3	17	3
19	535	3	4	7	2
20	635	15	22	37	2
21	546	38	6	44	2
22	757	2	3	5	6
23	633	19	3	22	9

number of clusters in each dataset. This procedure preserves dataset diversity across sources while ensuring a controlled and comparable sample size in evaluation.

To characterize task difficulty, we stratify datasets according to empirical clustering performance measured by ARI. Let $\max_m \mathrm{ARI}_m(\mathcal{D})$ denote the maximum ARI achieved on dataset $\mathcal{D}$ across

Table 10: **Statistics of OpenML datasets used in the real-world benchmark**.

ID	# Instances	# Numerical features	# Categorical features	Dimension	# Classes
16	2000	64	0	64	10
18	2000	6	0	6	10
22	2000	47	0	47	10
51	294	6	7	13	2
53	270	13	0	13	2
481	209	7	1	8	2
694	310	8	0	8	9
721	200	10	0	10	2
733	209	6	0	6	2
756	159	15	0	15	2
796	209	6	1	7	2
820	235	12	0	12	2
840	205	17	8	25	2
854	158	5	2	7	2
1063	522	21	0	21	2
1495	250	0	6	6	2
1510	569	30	0	30	2
1523	310	6	0	6	3
40496	500	7	0	7	10
40705	959	42	2	44	2
42585	344	4	2	6	3
40981	690	6	8	14	2

Table 11: **Statistics of UCI datasets used in the real-world benchmark**.

Name	# Instances	# Numerical features	# Categorical features	Dimension	# Classes
haberman	306	2	1	3	2
banknote	1372	4	0	4	2
iris	150	4	0	4	3
thyroid	215	5	0	5	3
ecoli	336	7	0	7	8
seeds	210	7	0	7	3
pima	768	8	0	8	2
breast	277	9	0	9	2
wisc	699	9	0	9	2
glass	214	9	0	9	6
wine	178	13	0	13	3
heart	303	13	0	13	2
vote	435	16	0	16	2
zoo	101	16	0	16	7
vehicle	846	18	0	18	4
segment	2310	19	0	19	7
wdbc	569	31	0	31	2
iono	351	34	0	34	2
dermatology	366	1	33	34	6
landsat	2000	36	0	36	6
biodeg	1055	41	0	41	2
spambase	4601	57	0	57	2

all compared methods m. Datasets are categorized as *Low Difficulty* if $\max_m \mathrm{ARI}_m(\mathcal{D}) \geq 0.4$, and *High Difficulty* if $0.1 \leq \max_m \mathrm{ARI}_m(\mathcal{D}) < 0.4$. Datasets with $\max_m \mathrm{ARI}_m(\mathcal{D}) < 0.1$ are excluded, as such cases provide little evidence of recoverable cluster structure and would confound interpretation of method comparisons. The threshold $\max_m \mathrm{ARI} \geq 0.4$ follows the ZEUS evaluation protocol, where only datasets above this separability level are included. Our benchmark subsumes

these ZEUS datasets and further extends the evaluation to more challenging regimes by incorporating datasets with lower but non-trivial ARI (0.1–0.4), thereby covering a broader spectrum of realistic clustering difficulty.

C.2 Baseline details

We first describe the baseline methods. Their acronyms are listed below together with brief descriptions.

- **AP**. This method corresponds to *Affinity Propagation*, which forms clusters by iteratively passing messages between pairs of samples until convergence to a set of exemplars. We use the implementation in the `scikit-learn` package with default parameter settings on all datasets.
- **KMeans**. This method corresponds to the classical k*-means* clustering algorithm, which partitions data into K clusters by minimizing the within-cluster sum of squared Euclidean distances. We use the `scikit-learn` implementation with default settings, where the number of clusters is specified according to the evaluation protocol.
- **GMM**. This method refers to the *Gaussian Mixture Model* with full covariance matrices, where each component is associated with its own general covariance matrix. Model parameters estimates are computed via the expectation–maximization (EM) algorithm. We use the `scikit-learn` implementation with default configuration unless otherwise specified.
- **S-GMM**. This method denotes a simplified Gaussian mixture model in which all components share the same identity covariance matrix. We adopt the implementation in the official implementation of ZEUS [22].
- **Agg**. This method corresponds to *Agglomerative Clustering*, a hierarchical bottom-up approach that iteratively merges clusters according to a linkage criterion. We use the `scikit-learn` implementation with default linkage and distance metric settings.
- **Spec/Spectral**. This method refers to *Spectral Clustering*, which constructs a similarity graph over samples and performs clustering in the space spanned by the leading eigenvectors of the graph Laplacian. We use the `scikit-learn` implementation with default affinity and eigen-solver settings.
- **Birch**. This method corresponds to *BIRCH* (Balanced Iterative Reducing and Clustering using Hierarchies), which incrementally builds a clustering feature tree to summarize the data and performs clustering based on this compressed representation. We use the `scikit-learn` implementation with default threshold and branching factor settings.
- **DBSCAN**. This method corresponds to *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)*, which groups together points that lie in high-density regions and labels low-density points as noise based on neighborhood radius and minimum density criteria. We use the `scikit-learn` implementation with default neighborhood and minimum-sample settings.
- **HDBSCAN**. This method refers to *Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)*, an extension of DBSCAN that constructs a hierarchy of density-based clusters and extracts a stable flat clustering based on cluster persistence. We use the standard `scikit-learn` library implementation with default parameter settings.
- **OPTICS**. This method corresponds to *Ordering Points To Identify the Clustering Structure (OPTICS)*, which produces an ordering of samples reflecting the underlying density-based cluster structure and enables extraction of clusters at multiple density levels. We use the `scikit-learn` implementation with default reachability and minimum-sample parameters.
- **MeanShift**. This method refers to *Mean Shift Clustering*, a non-parametric approach that seeks modes of a kernel density estimate by iteratively shifting points toward regions of higher density. We use the `scikit-learn` implementation with automatic bandwidth estimation and default settings.
- **ZEUS+X**. This class of methods uses the pretrained checkpoint provided by ZEUS [22] to extract feature embeddings, followed by applying a downstream clustering algorithm X in the embedding space. We consider X $\in \{k$-means, GMM, S-GMM$\}$, following the evaluation protocol adopted in ZEUS.

- **T+X**. The method is proposed in [19]. It follows an iterative pseudo-labeling scheme built on TabPFN v2. First, a subset of samples is drawn from the dataset, and a clustering algorithm X is applied to obtain initial cluster assignments. These samples and their cluster labels are then used as context to fine-tune TabPFN v2, which subsequently predicts cluster labels for all samples, producing a fully pseudo-labeled dataset. Next, a new subset is sampled from this pseudo-labeled dataset and again used as context to fine-tune TabPFN v2, which re-predicts cluster labels for the full dataset. This subsample–condition–predict procedure is repeated iteratively, progressively refining the pseudo labels. We implement this algorithm following the descriptions in the paper.

C.3 Data preprocessing details

Since the datasets contain both categorical and numerical features, preprocessing differs between `TabClustPFN` and the baseline methods.

For ZEUS and all baselines implemented in `scikit-learn`, categorical variables are converted to one-hot representations and concatenated with the numerical features along the feature dimension. In contrast, `TabClustPFN` does not apply explicit one-hot encoding. During both pretraining and evaluation, categorical variables are processed directly by the model's encoder, following the treatment used in TabICL.

For numerical features, we apply feature-wise standardization for all methods: each variable is centered by subtracting its sample mean and scaled by its sample standard deviation, resulting in zero mean and unit variance per feature. For ZEUS, we additionally follow their preprocessing protocol by applying a MinMax scaling step after standardization, mapping each numerical feature to the range $[0, 1]$.

C.4 Amortized clustering experiments details

Data generation for the amortized clustering comparison. We follow the common 2D synthetic setup in amortized clustering, but use a more diverse overlap-controlled generator to systematically vary cluster separation (see App. B.1 for details). Within each minibatch, different datasets may have different ground-truth numbers of clusters K. In addition, Gaussian components are sampled with full covariance matrices rather than being restricted to diagonal (or isotropic) covariances, yielding rotated and correlated clusters while keeping the input as an unordered set of points.

Training and evaluation protocol. We train and evaluate all methods on the same synthetic datasets: at each training step we sample a batch of datasets from the generator above, and evaluation is performed on independently sampled held-out datasets using a different random seed. We train for $1{,}000$ iterations with batch size 16 and evaluate for 100 iterations. For NCP and DAC, we use their default optimizer and learning-rate schedules as specified in the original papers; we refer readers to those works for the complete optimization details.

D Additional results

This section provides additional experimental results. We first report median ARI and NMI values on the evaluation sets (Sec. D.1) and ranks of methods with known K on synthetic data (Sec. D.2). We then provide a controlled comparison with Cluster-PFN (Sec. D.4), representation visualizations (Sec. D.3), statistical significance tests (Sec. D.6), pairwise real-world comparisons (Sec. D.8), cardinality calibration results (Sec. D.7), and real-world results grouped by knowledge of K (Sec. D.9). We additionally report synthetic distribution shifts and real-world feature-OOD evaluations in Sec. D.10.

D.1 Median ARI and NMI values grouped by levels of difficulty

In the main paper, we report median ARI and NMI *ranks* to account for varying dataset difficulty. Here, we additionally report the median ARI and NMI *values* of each method. As described in Sec. C.1, datasets are grouped into two levels of difficulty (low and high), enabling a more controlled comparison. Tables 12 and 13 present the corresponding results on synthetic and real-world benchmarks. Variants marked † use method-specific selection criteria:

- **DBSCAN**† maximizes density-based clustering validation (DBCV), $\sum_c \frac{|C_c|}{N} \frac{b_c - a_c}{\max(a_c, b_c)}$, where C_c are non-noise clusters, N counts all observations, and a_c and b_c denote within-cluster density sparseness and minimum between-cluster density separation, respectively. We search 41 log-spaced values of $\varepsilon \in [0.1, 10]$ and `min_samples` $\in \{3, 5, 10, 20\}$.
- **GMM**† selects $K \in \{2, \ldots, 10\}$ by minimizing the MAP approximation to integrated completed likelihood, $\mathrm{ICL}_{\mathrm{BIC}}(K) = \mathrm{BIC}(K) - 2\sum_i \log \max_k \tau_{ik}$, where τ_{ik} is the posterior component probability. We use full covariance matrices and one initialization per candidate.
- **Spectral**† selects $\hat{K} = \arg\max_{K \in \{2,\ldots,10\}} (\lambda_{K+1} - \lambda_K)$, where $\lambda_1 \leq \lambda_2 \leq \cdots$ are the eigenvalues of the normalized graph Laplacian constructed using an RBF affinity.

Table 12: **Comprehensive performance on synthetic datasets by difficulty**. Results show median [IQR] for ARI and NMI (higher is better), and k-MAE. † denotes method-specific selection criteria described in the text.

Method	ARI [IQR] ↑		NMI [IQR] ↑		k-MAE ↓	
	Low	High	Low	High	Low	High
TabClustPFN*	0.8348 [0.22]	0.1125 [0.10]	0.8012 [0.28]	0.1571 [0.14]	-	-
TabClustPFN	**0.8360 [0.24]**	0.1321 [0.08]	0.8046 [0.32]	0.1940 [0.10]	**0**	5
AP	0.0551 [0.10]	0.0243 [0.02]	0.3169 [0.28]	0.1440 [0.06]	44	45
Agg*	0.6586 [0.39]	0.1536 [0.07]	0.6513 [0.35]	0.1562 [0.01]	-	-
Agg^{+}	0.6081 [0.46]	0.1278 [0.09]	0.6006 [0.40]	0.1087 [0.04]	1	3
Birch*	0.6489 [0.41]	0.0886 [0.00]	0.6536 [0.36]	0.1422 [0.02]	-	-
Birch^{+}	0.5836 [0.48]	0.0628 [0.03]	0.5355 [0.41]	0.0947 [0.03]	1	3
DBSCAN	0.0000 [0.00]	0.0000 [0.00]	0.0000 [0.00]	0.0000 [0.02]	5	6
DBSCAN†	0.0129 [0.18]	0.0019 [0.00]	0.1215 [0.21]	0.0476 [0.02]	3	10
GMM*	0.6990 [0.44]	**0.1769 [0.08]**	0.7442 [0.36]	**0.1956 [0.02]**	-	-
GMM^{+}	0.7510 [0.48]	0.1538 [0.10]	0.7142 [0.41]	0.1618 [0.05]	1	4
GMM†	0.4189 [0.42]	0.1538 [0.10]	0.5094 [0.37]	0.1618 [0.05]	3	4
HDBSCAN	0.0000 [0.16]	-0.0071 [0.01]	0.0881 [0.25]	0.0056 [0.01]	3	5
KMeans*	0.6838 [0.49]	0.1688 [0.10]	0.7386 [0.39]	0.1883 [0.04]	-	-
KMeans^{+}	0.7350 [0.46]	0.1749 [0.09]	0.7054 [0.42]	0.1886 [0.04]	1	4
MeanShift	0.0000 [0.00]	-0.0002 [0.00]	0.0000 [0.00]	0.0018 [0.00]	4	4
OPTICS	0.0000 [0.01]	-0.0003 [0.00]	0.0110 [0.08]	0.0209 [0.02]	4	8
Spectral*	0.0682 [0.37]	0.1092 [0.11]	0.0867 [0.48]	0.1326 [0.09]	-	-
Spectral^{+}	0.0192 [0.38]	0.1100 [0.11]	0.0539 [0.49]	0.1143 [0.11]	2	4
Spectral†	0.2366 [0.45]	0.0162 [0.02]	0.2956 [0.44]	0.0506 [0.01]	3	3
T+Agg*	0.7879 [0.35]	0.0326 [0.07]	**0.8160 [0.33]**	0.1481 [0.03]	-	-
T+Agg^{+}	0.7326 [0.57]	-0.0085 [0.03]	0.6875 [0.45]	0.1077 [0.01]	1	**2**
T+GMM*	0.7651 [0.43]	0.1644 [0.07]	0.7389 [0.37]	0.1827 [0.02]	-	-
T+GMM^{+}	0.7711 [0.59]	0.1421 [0.09]	0.7013 [0.40]	0.1582 [0.04]	1	4
T+KMeans*	0.7413 [0.54]	0.1642 [0.08]	0.7217 [0.45]	0.1809 [0.03]	-	-
T+KMeans^{+}	0.7413 [0.54]	0.1572 [0.08]	0.6457 [0.45]	0.1734 [0.03]	1	**2**
T+Spec*	0.7939 [0.47]	0.1282 [0.05]	0.7659 [0.39]	0.1505 [0.01]	-	-
T+Spec^{+}	0.7850 [0.66]	0.1243 [0.05]	0.7131 [0.48]	0.1286 [0.13]	1	4
Zeus+GMM*	0.7624 [0.47]	0.0917 [0.06]	0.6914 [0.38]	0.1534 [0.13]	-	-
Zeus+GMM^{+}	0.3771 [0.44]	0.0908 [0.06]	0.4821 [0.30]	0.1785 [0.11]	6	5
Zeus+KMeans*	0.7696 [0.47]	0.0897 [0.06]	0.6939 [0.37]	0.1538 [0.13]	-	-
Zeus+KMeans^{+}	0.4041 [0.51]	0.0935 [0.06]	0.5035 [0.34]	0.1771 [0.12]	6	5
Zeus+S-GMM*	0.7684 [0.46]	0.0912 [0.06]	0.6920 [0.38]	0.1534 [0.13]	-	-
Zeus+S-GMM^{+}	0.5602 [0.54]	0.0876 [0.07]	0.6129 [0.35]	0.1781 [0.12]	6	5

In terms of ARI, `TabClustPFN` achieves the strongest overall performance in most settings, with the main exception being high-difficulty synthetic datasets. For NMI, `TabClustPFN` remains among the top-performing methods across nearly all groups, and stays within the top three even in its weakest regime (low-difficulty real datasets). Difficulty-stratified analysis reveals several trends. First, performance degradation on high-difficulty datasets is universal: even with oracle K, classical methods such as GMM* and KMeans* exhibit low ARI, indicating that the dominant challenge is recovering assignments under heavy cluster overlap and complex structure, rather than model selection alone. Second, in the practical unknown-K setting, `TabClustPFN` achieves the best or near-best ARI/NMI on low-difficulty synthetic datasets while accurately recovering K, demonstrating

good calibration of model complexity. On real datasets, TabClustPFN* attains the best ARI across difficulty levels, while the automatic-K variant remains competitive, suggesting graceful degradation under distribution shift. Third, the performance gap between oracle and inferred K is generally smaller for TabClustPFN than for classical pipelines, indicating reduced sensitivity to exact model order. Finally, in settings where ARI is not strictly best but NMI remains high, errors tend to preserve coarse cluster structure rather than producing highly fragmented partitions. Overall, these results suggest that the primary strength of TabClustPFN lies in stable assignment quality and robust K inference in realistic exploratory settings, rather than dependence on favorable parametric assumptions. Further gains may be achievable by broadening the pretraining prior or extending training.

Table 13: **Comprehensive performance on real-world benchmark by difficulty**. Results show median [IQR] for ARI and NMI (higher is better), and k-MAE. † denotes method-specific selection criteria described in the text.

Method	ARI [IQR] ↑		NMI [IQR] ↑		k-MAE ↓	
	Low	High	Low	High	Low	High
TabClustPFN*	**0.5754 [0.31]**	**0.1897 [0.21]**	0.5737 [0.34]	0.1614 [0.13]	-	-
TabClustPFN	0.4431 [0.29]	0.1801 [0.14]	0.5311 [0.28]	**0.1843 [0.16]**	1	1
AP	0.1217 [0.18]	0.0298 [0.05]	0.3306 [0.29]	0.1408 [0.13]	27	40
Agg*	0.5104 [0.27]	0.1404 [0.06]	0.5665 [0.32]	0.0786 [0.05]	-	-
Agg$^+$	0.5019 [0.37]	0.1359 [0.11]	0.4544 [0.34]	0.0915 [0.06]	**0**	2
Birch*	0.4157 [0.31]	0.1435 [0.08]	0.5132 [0.40]	0.0920 [0.07]	-	-
Birch$^+$	0.2980 [0.45]	0.1435 [0.07]	0.4245 [0.43]	0.0965 [0.08]	**0**	1
DBSCAN	0.0000 [0.20]	-0.0029 [0.02]	0.1158 [0.29]	0.0303 [0.06]	3	2
DBSCAN†	0.0691 [0.24]	0.0003 [0.10]	0.2059 [0.32]	0.0549 [0.07]	4	3
GMM*	0.4453 [0.28]	0.1313 [0.19]	0.5147 [0.38]	0.1068 [0.19]	-	-
GMM$^+$	0.3772 [0.40]	0.1430 [0.16]	0.4028 [0.38]	0.1156 [0.22]	**0**	1
GMM†	0.3407 [0.27]	0.1265 [0.15]	0.4222 [0.30]	0.1528 [0.22]	3	4
HDBSCAN	0.1663 [0.30]	0.0042 [0.04]	0.2628 [0.28]	0.0843 [0.11]	4	6
KMeans*	0.5106 [0.22]	0.1362 [0.13]	0.5485 [0.33]	0.1012 [0.20]	-	-
KMeans$^+$	0.5005 [0.32]	0.1179 [0.13]	0.4942 [0.34]	0.1168 [0.22]	**0**	1
MeanShift	0.0707 [0.31]	0.0365 [0.21]	0.1352 [0.35]	0.0535 [0.13]	5	8
OPTICS	0.0168 [0.04]	-0.0195 [0.04]	0.1788 [0.19]	0.0979 [0.09]	3	9
Spectral*	0.0578 [0.37]	0.0081 [0.06]	0.0828 [0.53]	0.0190 [0.04]	-	-
Spectral$^+$	0.0786 [0.38]	0.0055 [0.01]	0.1276 [0.44]	0.0142 [0.03]	1	**0**
Spectral†	0.3012 [0.43]	0.0751 [0.14]	0.2849 [0.42]	0.0628 [0.14]	4	4
T+Agg*	0.5119 [0.31]	0.0867 [0.15]	0.5536 [0.29]	0.0534 [0.13]	-	-
T+Agg$^+$	0.3200 [0.27]	0.0903 [0.07]	0.4120 [0.32]	0.1424 [0.17]	5	6
T+GMM*	0.4841 [0.27]	0.0859 [0.13]	0.5535 [0.25]	0.0701 [0.16]	-	-
T+GMM$^+$	0.2980 [0.25]	0.1063 [0.08]	0.4091 [0.32]	0.1249 [0.17]	5	6
T+KMeans*	0.5148 [0.29]	0.0919 [0.13]	0.5183 [0.29]	0.0712 [0.12]	-	-
T+KMeans$^+$	0.2741 [0.27]	0.0882 [0.12]	0.3913 [0.32]	0.1217 [0.19]	5	5
T+Spec*	0.4124 [0.26]	0.0024 [0.17]	0.5141 [0.30]	0.0510 [0.14]	-	-
T+Spec$^+$	0.2421 [0.25]	0.0489 [0.12]	0.3813 [0.32]	0.1037 [0.20]	6	8
Zeus+GMM*	0.5292 [0.43]	0.0501 [0.17]	0.6192 [0.40]	0.0713 [0.12]	-	-
Zeus+GMM$^+$	0.3155 [0.38]	0.0698 [0.07]	0.4377 [0.39]	0.1070 [0.08]	8	8
Zeus+KMeans*	0.5209 [0.42]	0.0700 [0.26]	0.6199 [0.39]	0.1126 [0.15]	-	-
Zeus+KMeans$^+$	0.3282 [0.36]	0.0615 [0.08]	0.4561 [0.39]	0.1089 [0.08]	8	8
Zeus+S-GMM*	0.5186 [0.38]	0.0834 [0.24]	**0.6289 [0.39]**	0.1242 [0.14]	-	-
Zeus+S-GMM$^+$	0.4145 [0.48]	0.0800 [0.07]	0.4670 [0.35]	0.1081 [0.40]	8	8

D.2 Median ARI and NMI values on the synthetic for methods with known K

In the main paper, we report median ARI and NMI ranks when the number of clusters K is inferred from the data. Here, we additionally evaluate the setting where the true K is provided to all methods. Tab. 14 reports the median ARI and NMI ranks on synthetic datasets under this oracle-K scenario. When the true number of clusters is given, TabClustPFN also achieves the best median ARI rank and NMI rank overall, indicating that its advantage is not solely due to accurate K inference but also to improved assignment quality.

Table 14: **Performance on synthetic datasets**. Median ARI/NMI ranks of methods with known K. Brackets show IQR. Bold and underline indicate best and second-best per column.

Method	ARI Rank [IQR] ↓	NMI Rank [IQR] ↓
TabClustPFN*	**1 [4]**	**3 [5]**
Agg*	8 [4]	9 [4]
Birch*	8 [5]	9 [4]
GMM*	6 [7]	5 [5]
KMeans*	6 [6]	6 [6]
Spectral*	13 [2]	13 [1]
T+Agg*	<u>4 [5]</u>	**3 [5]**
T+GMM*	6 [5]	6 [6]
T+KMeans*	6 [6]	6 [7]
T+Spec*	<u>4 [5]</u>	<u>4 [6]</u>
Zeus+GMM*	7 [6]	6 [5]
Zeus+KMeans*	6 [7]	6 [6]
Zeus+S-GMM*	6 [7]	6 [5]

D.3 Representation visualization on all synthetic datasets

In the main paper (Fig. 3), we presented representative examples to illustrate how different models shape the representation space. Here, we extend that analysis by providing t-SNE visualizations for *all* synthetic datasets used in our evaluation. Figures 10 and 11 show results for datasets generated from GMM priors and the ZEUS prior, respectively. The same protocol is followed as in the main text: we compare raw input features, pretrained TabICL embeddings, ZEUS representations, and `TabClustPFN` representations after the final projection and normalization. Points are colored by ground-truth labels, and for `TabClustPFN`, red stars denote learned cluster prototypes. This comprehensive view confirms that the geometric patterns highlighted in the main paper—such as the tendency of `TabClustPFN` to form more compact and prototype-aligned clusters, and the anisotropic or elongated structures often observed in ZEUS representations—are consistent across datasets rather than isolated cases. It also becomes evident that ZEUS tends to perform better when the raw feature space already exhibits blob-like cluster structure, and comparatively worse when such structure is absent.

D.4 Comparison with Cluster-PFN

While Cluster-PFN [3] shares the objective of utilizing in-context learning to predict both cluster cardinality and assignments, its scope is restricted to a narrower low-dimensional GMM regime (2 to 5 feature dimensions). Consequently, to ensure a rigorous and fair evaluation, we compare the two approaches both methodologically and empirically under this specific controlled setting.

Methodological comparison. Table 15 summarizes the main differences between Cluster-PFN and `TabClustPFN`. Although both methods follow the PFN paradigm, they differ substantially in the prior, the assignment objective, the assignment architecture, and the mechanism for cardinality inference. Cluster-PFN is primarily designed around finite GMM priors and resolves label-switching through deterministic relabeling before applying a supervised cross-entropy loss. In contrast, `TabClustPFN` is pretrained on a broader prior family combining controlled-overlap GMMs and the ZEUS prior, and optimizes a permutation-invariant SoftARI objective. Architecturally, `TabClustPFN` introduces explicit cluster prototypes and iterative prototype–data cross-attention in the Partition Inference Network (PIN), while the Cardinality Inference Network (CIN) estimates $p(K \mid X)$ from assignment Gram matrices produced across candidate values of K. These differences make the cluster structure more explicit and provide a different route for scaling amortized clustering beyond very low-dimensional GMMs.

Controlled GMM evaluation. To provide a fair empirical comparison within the regime where Cluster-PFN is designed to operate, we follow its low-dimensional GMM setting and evaluate all methods on newly generated held-out datasets from a GMM prior. For each feature dimension

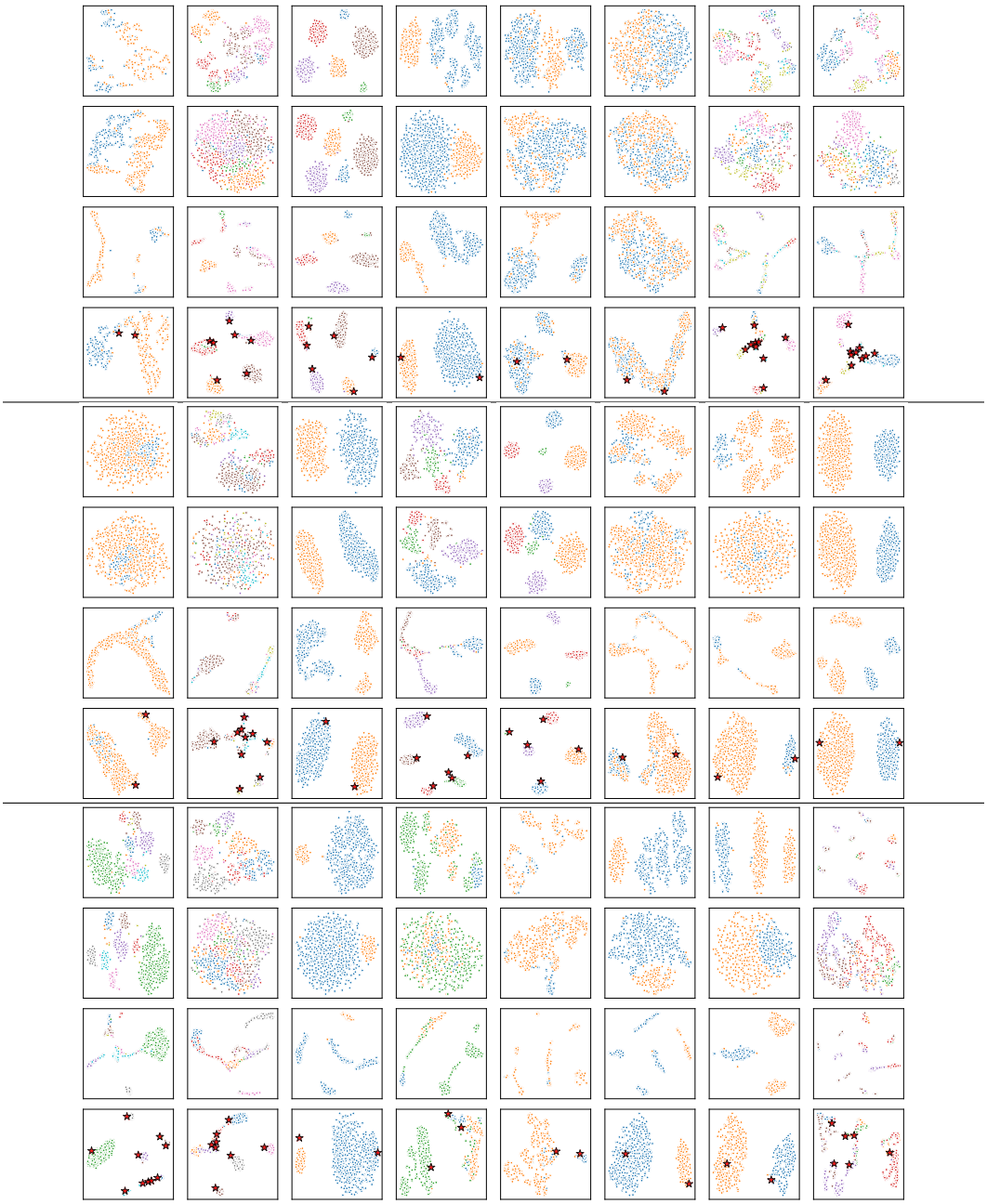

Figure 10: **Representation t-SNE visualizations on GMM prior**. Rows are organized in groups of four; within each group, columns correspond to different datasets. From top to bottom, rows display raw features, TabICL encoder outputs, ZEUS representations, and TabClustPFN representations.

$D \in \{2, 3, 4, 5\}$, we generate 320 independent datasets, resulting in 1,280 datasets in total. Each dataset contains $N = 500$ observations and has an unknown number of clusters with maximum cluster count $K_{\max} = 10$. All methods are evaluated in the unknown-K setting, where the true cluster number is not provided at test time. In addition to Cluster-PFN and `TabClustPFN`, we include GMM$^+$ as a classical baseline, where the number of mixture components is selected by maximizing the silhouette score. This baseline is particularly relevant because the test data are themselves generated from a GMM prior, making the evaluation favorable to GMM-based methods.

Figure 12 reports ARI, NMI, and k-MAE across feature dimensions. At $D = 2$, the three methods perform comparably, which is expected since this is the simplest setting and is well aligned with

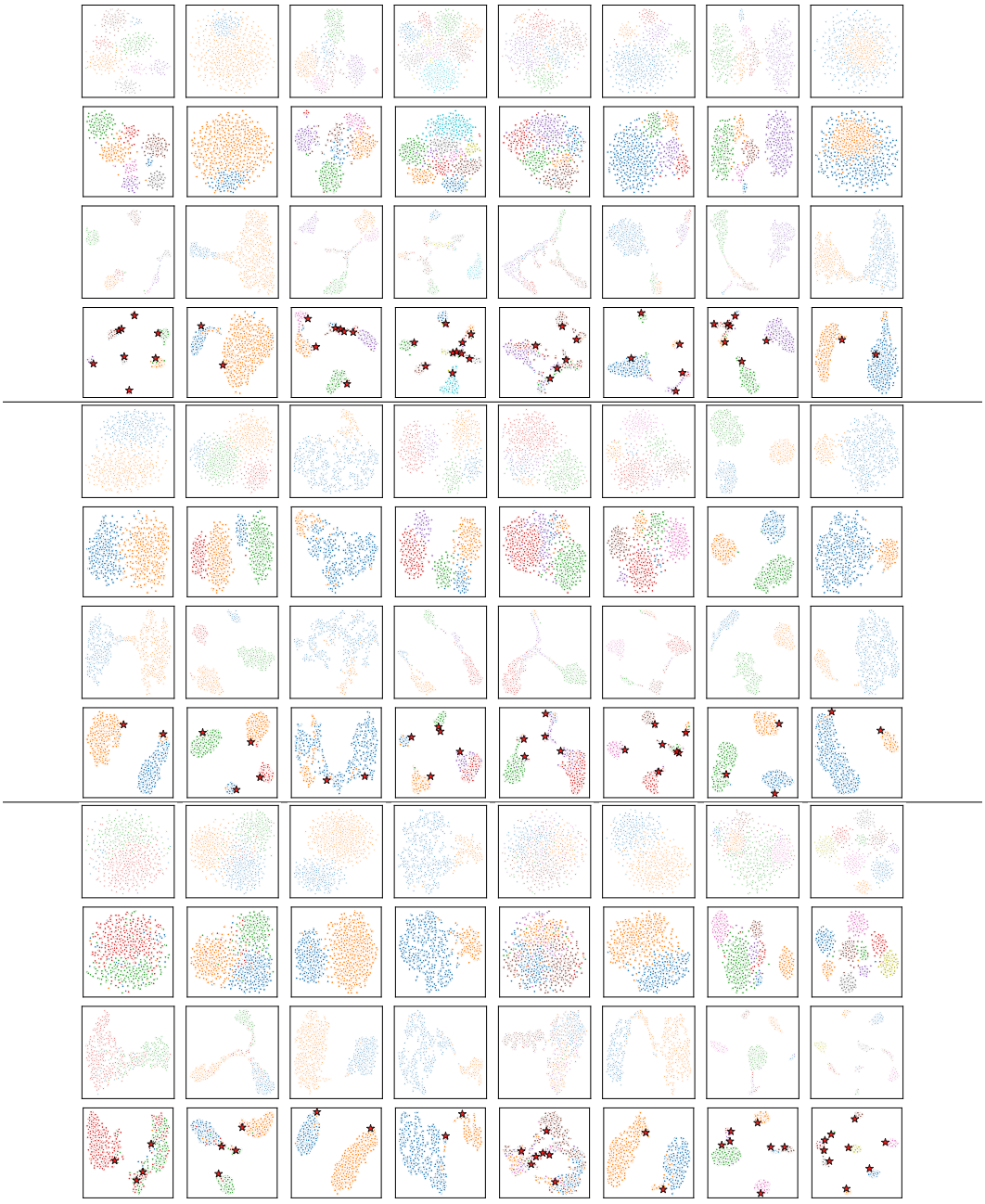

Figure 11: **Representation t-SNE visualizations on ZEUS prior**. Rows are organized in groups of four; within each group, columns correspond to different datasets. From top to bottom, rows display raw features, TabICL encoder outputs, ZEUS representations, and TabClustPFN representations.

Table 15: Methodological comparison between Cluster-PFN and `TabClustPFN`. Both methods perform amortized clustering, but differ in their prior, objective, architecture, and cardinality-inference mechanism.

Aspect	**Cluster-PFN**	`TabClustPFN`
Prior	Finite GMM prior in a low-dimensional setting ($D \leq 5$).	Controlled-overlap GMM prior combined with the ZEUS prior, covering both Gaussian and nonlinear non-Gaussian cluster geometries ($D \leq 64$).
Cluster-assignment objective	Uses deterministic relabeling to break label permutation symmetry, followed by cross-entropy.	Optimizes a permutation-invariant SoftARI loss, avoiding deterministic relabeling conventions that can be sensitive to scaling and dimensionality.
Cluster-assignment architecture	Uses standard Transformer layers followed by an MLP output head.	Uses a TabICL encoder followed by an iterative prototype–data cross-attention decoder in PIN.
Cluster-assignment mechanism	Outputs membership probabilities directly from point embeddings.	Maintains K cluster prototypes and computes assignments via cosine similarities between point representations and prototypes.
Cardinality-inference architecture	Constructs a special object ρ from a fixed all--1 vector, concatenates it with dataset embeddings, and maps the decoded representation of ρ to $p(K \mid X)$.	Uses CIN to map assignment Gram matrices $\{P_K^\top P_K\}_{K=2}^{K_{\max}}$ to a posterior over K.
Conditioning on K	Conditions the model through an embedding of the candidate value of K.	Conditions PIN by selecting the first K cluster prototypes, so the active prototypes directly define the candidate cardinality.

the assumptions of both Cluster-PFN and GMM$^+$. As the feature dimension increases, however, the performance of Cluster-PFN decreases in both clustering quality and cardinality estimation. In contrast, `TabClustPFN` remains stable and outperforms Cluster-PFN for all dimensions $D > 2$ across ARI, NMI, and k-MAE. The gap is most significant for $D = 4$ and $D = 5$, where `TabClustPFN` achieves substantially higher ARI and NMI while also producing more accurate estimates of the number of clusters.

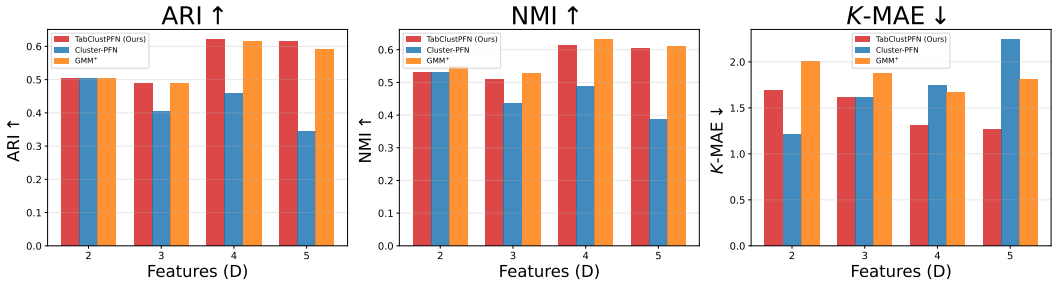

Figure 12: Comparison with Cluster-PFN under the low-dimensional GMM regime. All methods are evaluated in the unknown-K setting. Higher ARI/NMI and lower k-MAE indicate better performance. `TabClustPFN` consistently outperforms Cluster-PFN for $D > 2$.

D.5 Comparison with adaptive-cardinality baselines

CIN infers a posterior over a fixed set of candidate cardinalities, with $K_{\max} = 10$ in the current checkpoint. To examine this design trade-off, we compare with DP-means [16] and Dip-means [15], which can adapt the number of clusters without a fixed upper bound. This comparison evaluates clustering quality within the benchmark regime rather than extrapolation to $K > 10$.

Experimental setup. We reuse the 49 synthetic and 44 real-world datasets and sampled observations from Sec. 4 (seed 42). Both baselines follow the same preprocessing protocol as the classical baselines (App. C.3). TabClustPFN uses the unchanged checkpoint under unknown K. We compare package defaults with variants (†) selected using silhouette scores without ground-truth labels:

- **DP-means** (`pdc-dp-means` v0.0.8). Defaults use $\delta = 1$ and eight initial clusters. DP-means† searches δ over distinct positive quantiles $\{0.05, 0.10, \ldots, 0.95\}$ of pairwise squared Euclidean distances, using one initial cluster, five initializations, and at most 100 iterations.
- **Dip-means** (`ClustPy` v0.0.2). Defaults start from one cluster with significance level 0.001. Dip-means† searches significance levels $\{0.001, 0.005, 0.01, 0.05\}$.

Table 16: **Comparison with adaptive-cardinality baselines under unknown K.** ARI and NMI are reported as median [IQR], and k-MAE as the median absolute cardinality error. Unmarked baselines use package defaults; † denotes silhouette-selected hyperparameters.

Method	Synthetic (49 datasets)			Real-world (44 datasets)		
	ARI ↑	NMI ↑	k-MAE ↓	ARI ↑	NMI ↑	k-MAE ↓
TabClustPFN	**0.8308 [0.39]**	**0.8000 [0.42]**	**0**	**0.3853 [0.42]**	**0.4237 [0.36]**	**1**
DP-means	0.0552 [0.08]	0.3484 [0.29]	302	0.0351 [0.10]	0.2767 [0.25]	138
DP-means†	0.2892 [0.78]	0.4340 [0.67]	3	0.2774 [0.41]	0.3424 [0.36]	2
Dip-means	0.0000 [0.00]	0.0000 [0.00]	3	0.0000 [0.24]	0.0000 [0.44]	**1**
Dip-means†	0.0000 [0.00]	0.0000 [0.00]	4	0.0558 [0.39]	0.1245 [0.48]	**1**

Results and design trade-off. Table 16 shows that TabClustPFN achieves higher median ARI and NMI than both baselines under either configuration on both benchmarks. It also has the lowest median k-MAE, tied with Dip-means on real-world data. Thus, allowing cluster counts beyond CIN's support does not by itself yield better clustering within the evaluated regime. CIN additionally provides a posterior over K, while the PIN–CIN decomposition supports inference conditioned on a user-specified K. The trade-off is that the current checkpoint cannot infer $K > K_{\max}$; extending this support remains future work.

D.6 Statistical significance tests

We further assess whether the observed performance gains are statistically reliable across datasets. For each baseline method b and each evaluation dataset s, we compute the paired ARI difference

$$\Delta_b(s) = \mathrm{ARI}_{\texttt{TabClustPFN}}(s) - \mathrm{ARI}_b(s).$$

Thus, $\Delta_b(s) > 0$ indicates that `TabClustPFN` achieves a higher ARI than baseline b on dataset s. All significance tests are conducted on these per-dataset paired differences, separately for the synthetic benchmark and the real-world benchmark.

We use two complementary paired tests. First, we apply the one-sided paired t-test with alternative hypothesis

$$H_1 : \mathbb{E}[\Delta_b] > 0,$$

which tests whether `TabClustPFN` has a larger mean ARI than baseline b across datasets. The paired t-test is commonly used for paired algorithm comparisons, but it relies on a normality assumption for the paired differences. Since clustering scores are bounded, e.g., ARI lies in $[-1, 1]$ and NMI lies in $[0, 1]$, their empirical distributions need not be normal. We therefore also report the one-sided Wilcoxon signed-rank test, a non-parametric paired test that does not require normality of the paired

differences and tests whether the distribution of Δ_b is shifted toward positive values. This follows the recommendation of Demšar [6] to use non-parametric tests when comparing learning algorithms over multiple datasets.

In addition to pairwise tests, we report critical-difference (CD) diagrams based on average ARI ranks. For each dataset, all methods are ranked according to ARI, with lower rank indicating better performance. The CD diagram visualizes the average rank of each method and the critical difference required for two average ranks to be considered statistically distinguishable under the corresponding rank-based post-hoc comparison. We use CD diagrams as a complementary multi-method, rank-based view of the results, while the paired tests above directly compare `TabClustPFN` with each baseline in terms of ARI differences.

Paired t-tests. Figure 13 shows the distributions of $\Delta_b(s)$ and the one-sided paired t-test p-values. On the synthetic benchmark, `TabClustPFN` achieves positive paired differences against all baselines, and all paired t-tests reject the null hypothesis at the 0.05 level. Most comparisons are highly significant with $p < 0.001$. On the real-world benchmark, `TabClustPFN` also achieves positive median ARI differences against the vast majority of baselines. The paired t-tests show significant improvements over most baselines; the only non-significant comparisons are against Agg$^+$ and KMeans$^+$, which are the strongest classical competitors in this benchmark. Importantly, non-significance in these two comparisons does not indicate that the baselines outperform `TabClustPFN`; rather, the observed paired differences are not large enough to reject the null hypothesis under the paired t-test. `TabClustPFN` still achieves higher ARI on the majority of real-world datasets.

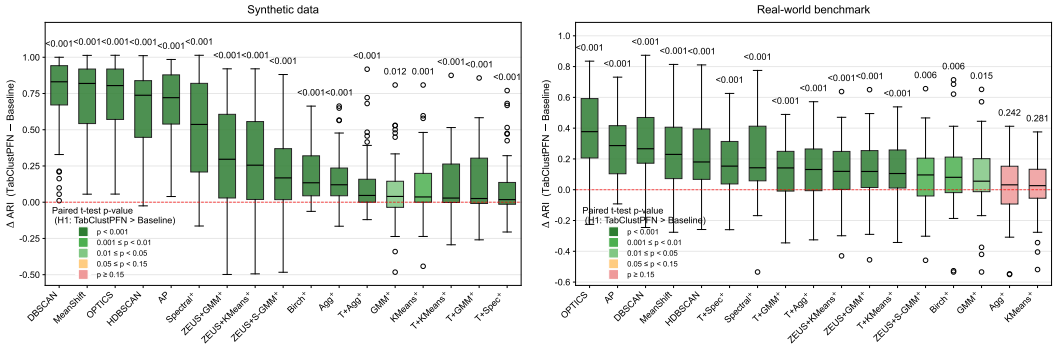

Figure 13: Paired t-test results based on per-dataset ARI differences $\Delta_b = \mathrm{ARI}_{\texttt{TabClustPFN}} - \mathrm{ARI}_b$. The dashed red line denotes zero difference. Positive values indicate that `TabClustPFN` outperforms the corresponding baseline. Colors indicate the one-sided paired t-test p-value under the alternative hypothesis $H_1 : \mathrm{ARI}_{\texttt{TabClustPFN}} > \mathrm{ARI}_b$.

Wilcoxon signed-rank tests. Figure 14 reports the corresponding one-sided Wilcoxon signed-rank tests. The results are consistent with the paired t-tests while avoiding the normality assumption on the paired differences. On the synthetic benchmark, `TabClustPFN` significantly outperforms all baselines. On the real-world benchmark, the Wilcoxon test yields the same qualitative conclusion and supports significant improvements over most baselines.

Critical-difference diagrams. Figure 15 provides a complementary rank-based comparison using critical-difference (CD) diagrams. On both the synthetic and real-world benchmarks, `TabClustPFN` achieves the best average ARI rank, with average ranks of 3.80 and 4.61, respectively. This ranking-based view is consistent with the paired tests above and further supports the overall advantage of `TabClustPFN` across benchmarks.

D.7 Calibration of cardinality inference

We evaluate the calibration of the cardinality posterior $\hat{p}(K \mid X)$ predicted by CIN. Since $\hat{p}(K \mid X)$ is a categorical distribution over candidate cluster cardinalities, we assess calibration through set-valued coverage. Given a nominal level α, we sort candidate cardinalities in descending order of $\hat{p}(K \mid X)$

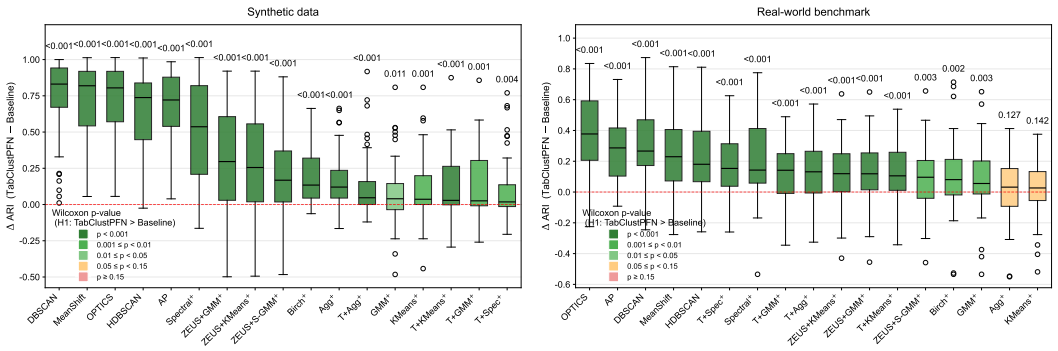

Figure 14: Wilcoxon signed-rank test results based on per-dataset ARI differences $\Delta_b = \mathrm{ARI}_{\texttt{TabClustPFN}} - \mathrm{ARI}_b$. Positive values indicate that `TabClustPFN` outperforms the corresponding baseline. Colors indicate the one-sided Wilcoxon signed-rank test p-value under the alternative hypothesis $H_1 : \mathrm{ARI}_{\texttt{TabClustPFN}} > \mathrm{ARI}_b$.

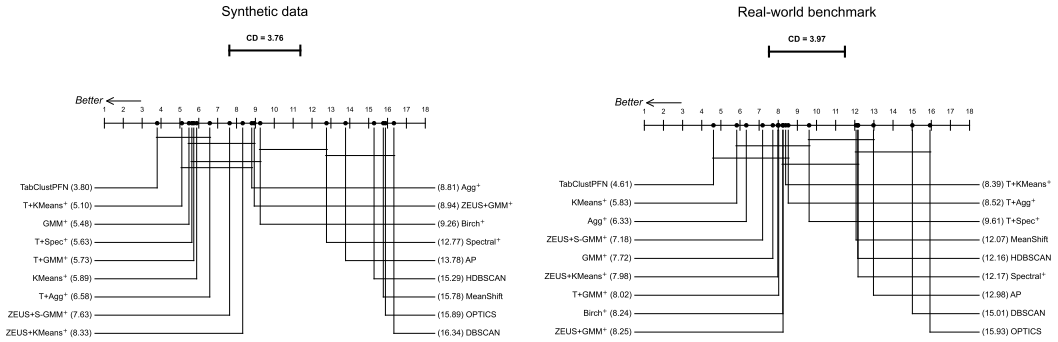

Figure 15: Critical-difference (CD) diagrams based on ARI ranks. Lower average rank indicates better performance.

and construct the smallest prediction set whose cumulative posterior mass reaches α:

$$\widehat{\mathcal{K}}_\alpha(X) = \{K_{(1)}, \ldots, K_{(m_\alpha)}\}, \qquad m_\alpha = \min\left\{ m : \sum_{\ell=1}^{m} \hat{p}(K_{(\ell)} \mid X) \geq \alpha \right\},$$

where $\hat{p}(K_{(1)} \mid X) \geq \hat{p}(K_{(2)} \mid X) \geq \cdots$. We then compute the empirical coverage

$$\frac{1}{S} \sum_{s=1}^{S} \mathbf{1}\left\{ K_s^* \in \widehat{\mathcal{K}}_\alpha(X_s) \right\},$$

where K_s^* is the ground-truth cluster cardinality for dataset s. A well-calibrated posterior should have empirical coverage close to the nominal level, while systematic undercoverage would indicate overconfidence.

Cardinality calibration is considered instead of point-level assignment calibration, since the latter is not directly well-defined in clustering. Since cluster labels are identifiable only up to permutation, assessing assignment calibration would require a post-hoc matching between predicted clusters and ground-truth labels, making the notion of a "correct" label dependent on a dataset-level permutation. By contrast, the cluster cardinality K is label-free and comparable across datasets.

We therefore evaluate set-valued calibration of $\hat{p}(K \mid X)$ on 3,200 held-out datasets generated from the GMM prior, covering feature dimensions from 2 to 64, sample sizes from 500 to 1000, and maximum cluster count $K_{\max} = 10$.

D.8 Pairwise win-rate comparison on real-world benchmark

To compare methods beyond average metrics, we visualize pairwise win rates across datasets in Fig. 16. The heatmap entry at row i and column j denotes the percentage of datasets on which

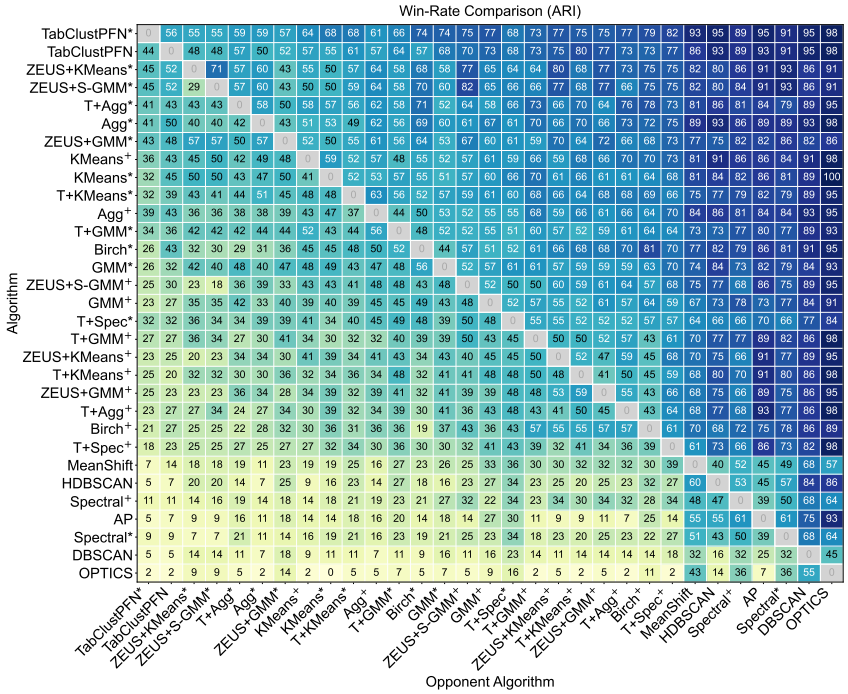

Figure 16: **Pairwise win rate comparison on real-world benchmark**. Each entry shows the percentage of datasets on which the row method achieves a higher ARI than the column method.

method i achieves a higher ARI than method j. Values greater than 50 therefore indicate that method i outperforms method j on the majority of datasets, while values below 50 indicate the opposite. Diagonal entries are undefined and omitted.

The figure compares all 31 methods, including those that assume known K and those that infer K from the data. As such, it provides a global view of relative robustness across datasets rather than isolating specific evaluation settings.

TabClustPFN* (with known K) exhibits win counts above 50 against all other methods, indicating consistent superiority across the majority of datasets. In contrast, TabClustPFN (with inferred K) does not outperform TabClustPFN*, ZEUS+Kmeans*, ZEUS+GMM*, and Agg*, but achieves higher win counts than the remaining baselines. This behavior is expected, as these four methods are provided with the true number of clusters, giving them additional information unavailable in the unknown-K setting.

D.9 Results grouped by information on the number of clusters

In the main paper, we report the median ARI rank of all methods on the real-world benchmark. To provide a more controlled comparison, we further group methods according to whether the number of clusters K is given or must be inferred. Fig. 17 reports median ARI ranks for methods with known K (top row) and methods with unknown K (bottom row). This stratified evaluation isolates performance differences under equal information conditions. In both settings, `TabClustPFN` achieves the best (lowest) median ARI rank among the compared methods, indicating consistently strong clustering performance given the same level of prior knowledge about K.

We additionally plot inference time against median ARI rank. These scatter plots illustrate the joint relationship between computational cost and clustering quality. `TabClustPFN` lies in the favorable region of this space, achieving the best clustering performance (lowest median ARI rank) while

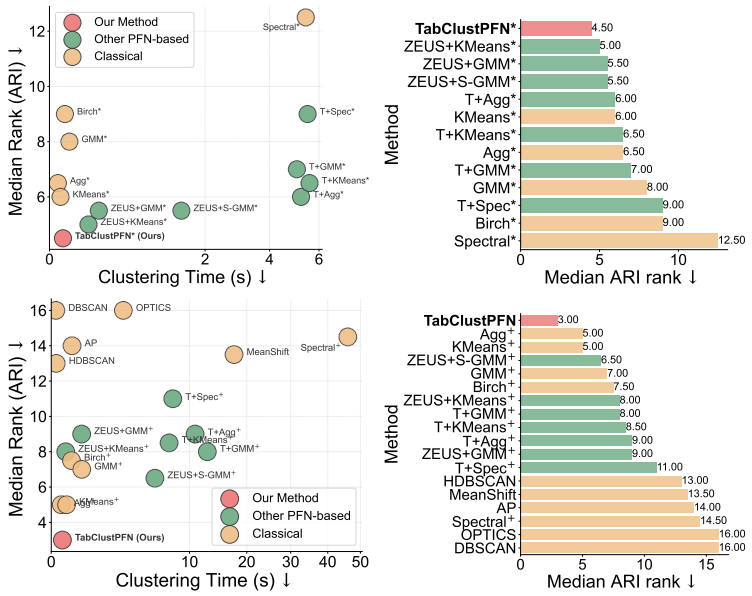

Figure 17: **Runtime and median ARI rank grouped by knowledge of K on real-world benchmark**. Top row: methods with known K. Bottom row: methods with unknown K.

requiring substantially less inference time than computationally intensive baselines such as Spectral clustering, and remaining more efficient than other PFN-based approaches.

D.10 Out-of-distribution generalization

D.10.1 Synthetic distribution shifts

Experimental setup. We use the main benchmark's unknown-K evaluation protocol and preprocessing (Sec. 4; App. C.3). Both experiments use $N = 1000$, $K = 3$, and $D \in \{2, 10, 30, 50\}$. Component centers are drawn uniformly from $[-5, 5]^D$.

(i) **Cluster-size imbalance.** We sample isotropic Gaussian components with covariance $s_k I$, $s_k \sim \mathcal{U}(0.3, 0.8)$, and target sizes proportional to $(r, 1, 1)$ for $r \in \{1, 2, 5, 10, 20, 50, 100\}$. Integer sizes are adjusted to sum to 1000.

(ii) **Heavy-tailed subpopulations.** We sample approximately equal-sized Student-t components with degrees of freedom $\nu \in \{1, 2, 3, 5, 10, 30, 1000\}$. Samples follow $x = \mu_k + \sigma_k z / \sqrt{v/\nu}$, where $z \sim \mathcal{N}(0, I_D)$, $v \sim \chi^2_\nu$, and $\sigma_k \sim \mathcal{U}(0.5, 1.0)$. Smaller ν gives heavier tails; $\nu = 1000$ approximates Gaussian components.

Results. Figure 18 compares methods across shift conditions at $D = 2$, where heavy-tailed clusters with small ν are particularly challenging. Figure 19 summarizes the trend across dimensions: TabClustPFN's median ARI approaches one for both shift families at higher dimensions.

D.10.2 Real-world OOD evaluation

Datasets and evaluation. Table 17 lists the 6 additional datasets selected from OpenML-CC18, TabArena, and the AutoML benchmark. Dataset selection and preprocessing follow Apps. C.1 and C.3, within the OOD range specified in the main text. We compare with KMeans$^+$, Agg$^+$, HDBSCAN, and ZEUS+KMeans$^+$ to cover centroid-based, hierarchical, density-based, and PFN-based clustering, respectively. The reported TabClustPFN results use direct inference without dimensionality reduction.

Results. Table 18 reports absolute metrics for the selected collection. TabClustPFN has the highest median ARI/NMI and lowest median cardinality error, with mean runtime close to KMeans$^+$. The broad IQRs indicate substantial variation across datasets.

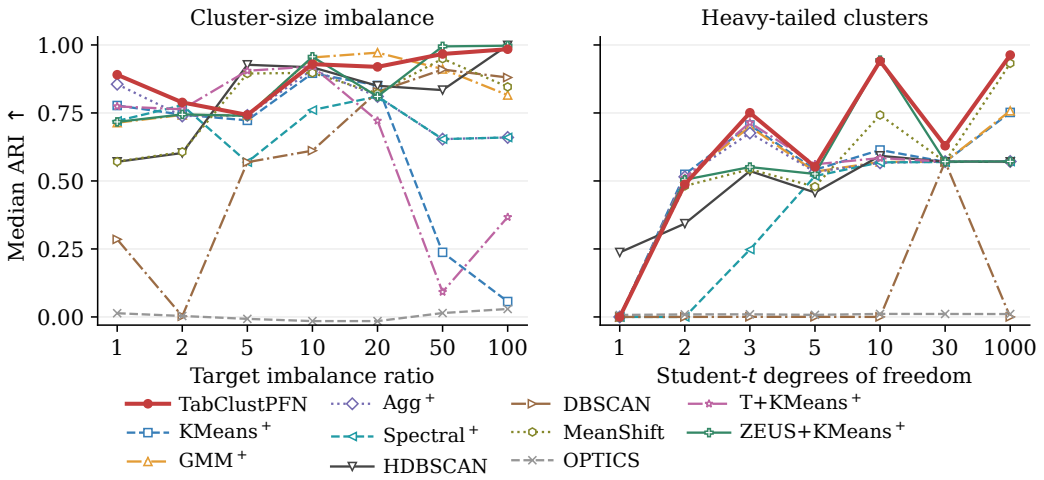

Figure 18: **Synthetic distribution shifts at $D = 2$.** Left: cluster-size imbalance. Right: heavy-tailed subpopulations. Each point reports median ARI ten data-generation seeds (0–9) for one condition under unknown K.

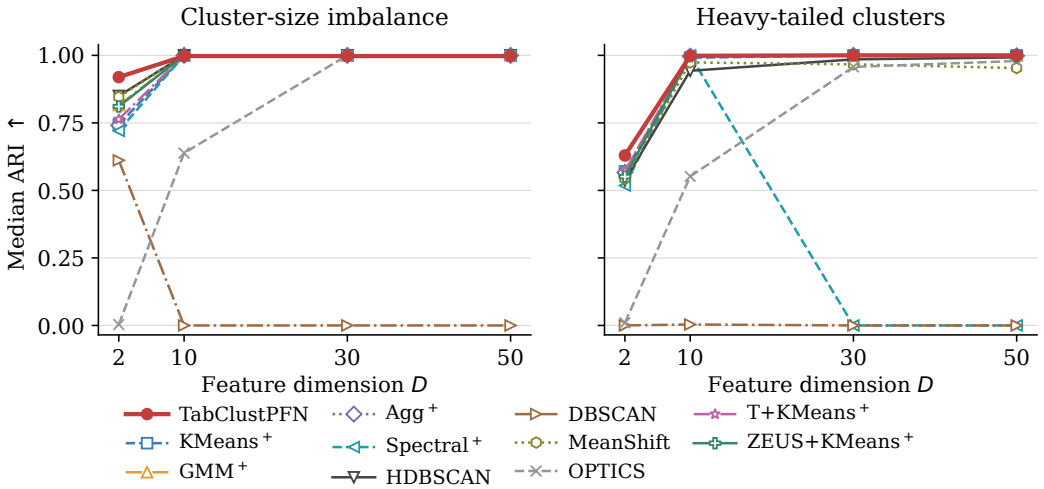

Figure 19: **Synthetic distribution shifts across feature dimensions.** For each method and dimension, we first compute median ARI ten data-generation seeds (0–9) within each condition, then take the median across the seven imbalance ratios (left) or seven degrees of freedom (right).

Table 17: **The 6 real-world OOD datasets.** OpenML IDs identify the exact datasets. N and D are the original sample and feature counts.

OpenML ID	Dataset	N	D	K
458	analcatdata_authorship	841	70	4
12	mfeat-factors	2000	216	10
46980	MIC	1699	111	8
14	mfeat-fourier	2000	76	10
41143	jasmine	2984	144	2
40966	MiceProtein	1080	77	8

Table 18: **Real-world OOD results on the 6 datasets in Table 17.** Median [IQR] ARI/NMI, median k-MAE, and mean clustering time. Bold marks the best result in each column.

Method	ARI ↑	NMI ↑	k-MAE ↓	Time (s) ↓
`TabClustPFN`	**0.200 [0.371]**	**0.352 [0.462]**	**1**	0.124
ZEUS+KMeans$^+$	0.165 [0.441]	0.327 [0.487]	5	1.363
KMeans$^+$	0.183 [0.469]	0.229 [0.523]	4	**0.106**
Agg$^+$	0.155 [0.470]	0.289 [0.545]	4	1.792
HDBSCAN	0.083 [0.064]	0.141 [0.319]	7	1.016

E Ablation study details

This section provides additional ablation results. Details of the baselines used in the PIN loss ablation are given in Sec. E.1, followed by the PIN decoder ablation (Sec. E.2) and further analysis of the CIN loss (Sec. E.3).

E.1 Different losses in PIN loss ablation

SoftARI is the default objective for PIN (Eq. (2)). To evaluate its effectiveness, we keep the PIN architecture, training protocol, and priors fixed, and replace only the training objective with several alternatives. Throughout, for a dataset with N observations and ground-truth cluster labels $\{z_i\}_{i=1}^N$ ($z_i \in \{1, \dots, K\}$), PIN produces logits $\ell_i \in \mathbb{R}^K$ and soft assignments $P_{ik} = \mathrm{softmax}(\ell_i)_k$. Let $Z \in \{0,1\}^{N \times K}$ be the one-hot encoding of $\{z_i\}$, i.e., $Z_{ij} = \mathbb{1}[z_i = j]$. We define the soft confusion matrix

$$M = P^\top Z \in \mathbb{R}^{K \times K}, \quad M_{kj} = \sum_{i=1}^{N} P_{ik} Z_{ij}.$$

Intuitively, M_{kj} is the total probability mass assigned to predicted cluster k by samples whose true label is j. Using P, instead of hard assignments from $\arg\max$, keeps M differentiable w.r.t. the logits, enabling gradient-based optimization.

A central challenge in clustering is label switching: relabeling predicted clusters corresponds to permuting the columns of P, which in turn permutes the rows of the soft confusion matrix M. An objective is permutation-invariant if it remains unchanged under such reindexing of M (e.g., by depending only on row/column-order-agnostic statistics) or if it explicitly optimizes over the best reindexing (e.g., via matching or a differentiable relaxation). We highlight this property for each ablated objective below.

E.1.1 SoftNMI loss

We compute a differentiable variant of normalized mutual information (SoftNMI) between the predicted partition and the ground-truth labels using the soft confusion matrix M. We first form the empirical joint distribution and marginals

$$\hat{p}_{kj} = \frac{M_{kj}}{N}, \qquad \hat{p}_k = \sum_{j=1}^{K} \hat{p}_{kj}, \qquad \hat{p}_j = \sum_{k=1}^{K} \hat{p}_{kj}.$$

The corresponding entropies are

$$H(\hat{p}_k) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k, \qquad H(\hat{p}_j) = -\sum_{j=1}^{K} \hat{p}_j \log \hat{p}_j,$$

and the mutual information is

$$I \;=\; \sum_{k=1}^{K} \sum_{j=1}^{K} \hat{p}_{kj} \log \frac{\hat{p}_{kj}}{\hat{p}_k \hat{p}_j}.$$

We adopt the arithmetic-mean normalization

$$\mathrm{SoftNMI} \;=\; \frac{2I}{H(\hat{p}_k) + H(\hat{p}_j)}.$$

The resulting training objective is

$$\mathcal{L}_{\mathrm{SoftNMI}} \;=\; 1 - \mathrm{SoftNMI}.$$

This objective is permutation-invariant: permuting predicted cluster indices only reorders the index k in the sums above, leaving the value unchanged.

E.1.2 Matching+CE loss

We consider a matching-based baseline that resolves label switching by explicitly selecting the best one-to-one reindexing between predicted clusters and ground-truth labels. Using the soft confusion matrix $M \in \mathbb{R}^{K\times K}$ as the matching score matrix, we compute a permutation matrix

$$\Pi^\star \in \arg\max_{\Pi\in\mathcal{P}_K} \langle \Pi, M\rangle_F,$$

where $\mathcal{P}_K = \{\Pi \in \{0,1\}^{K\times K} : \Pi\mathbf{1} = \mathbf{1}, \Pi^\top\mathbf{1} = \mathbf{1}\}$ and $\langle A, B\rangle_F = \sum_{k,j} A_{kj}B_{kj}$. Let $\pi^\star$ be the permutation induced by $\Pi^\star$ (i.e., $\Pi^\star_{k,j} = 1 \Leftrightarrow k = \pi^\star(j)$). We then remap each ground-truth label Z_i to a target predicted-cluster index $\tilde{Z}_i = \pi^\star(Z_i)$, and apply standard cross-entropy on the logits:

$$\mathcal{L}_{\text{Match+CE}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{CE}(\ell_i, \tilde{Z}_i).$$

Permutation invariance comes from the maximization over π: any reindexing of predicted clusters corresponds to permuting rows of M, but the optimal matching (and the resulting best-achievable agreement) is unchanged after re-solving the assignment. In practice, the matching step is discrete and is treated as a stop-gradient operation; gradients flow only through the cross-entropy term given the matched targets.

E.1.3 Matching+SoftAcc loss

To avoid the non-differentiability of discrete matching in $\mathcal{P}_K$, we replace $\Pi^\star$ with a differentiable relaxation $\Pi \in [0,1]^{K\times K}$ obtained via Sinkhorn normalization, and define a soft matched agreement by $\langle \Pi, M\rangle_F$, which reduces to $\langle \Pi^\star, M\rangle_F$ when Π is a hard permutation matrix.

We relax $\mathcal{P}_K$ to its convex hull, i.e., the Birkhoff polytope of doubly-stochastic matrices $\mathcal{B}_K = \{\Pi \geq 0 : \Pi\mathbf{1} = \mathbf{1}, \Pi^\top\mathbf{1} = \mathbf{1}\}$. Intuitively, $\mathcal{B}_K$ contains "soft permutations": each row and column sums to one, so Π_{kj} can be interpreted as a fractional one-to-one alignment weight between predicted cluster k and true cluster j. We obtain such a Π by applying Sinkhorn-Knopp normalization [32] to the log-scaled affinities:

$$\Pi = \mathrm{Sinkhorn}\left(\frac{\log(M+\epsilon)}{\epsilon}\right) \in [0,1]^{K\times K},$$

which iteratively normalizes rows and columns to produce an approximately doubly-stochastic matrix. The temperature ϵ controls how "hard" the alignment is: as ϵ decreases, Π becomes increasingly peaked and approaches a permutation-like matrix, recovering a continuous analogue of hard matching while remaining differentiable.

Given Π, we define a soft matched accuracy as the expected aligned agreement under this soft one-to-one correspondence:

$$\mathrm{SoftAcc} = \frac{1}{N}\langle \Pi, M\rangle_F = \frac{1}{N}\sum_{k=1}^{K}\sum_{j=1}^{K} \Pi_{kj}M_{kj}, \quad \mathcal{L}_{\text{Match+SoftAcc}} = 1 - \mathrm{SoftAcc}.$$

This quantity is a natural relaxation of matched clustering accuracy: when Π is a hard permutation matrix, $\langle \Pi, M\rangle_F$ reduces to the sum of the entries of M on the permuted diagonal, i.e., the total correctly aligned mass (and dividing by N yields an accuracy-like normalization). When Π is soft, the same expression computes the agreement averaged over fractional matchings, hence a "soft" accuracy. Because both Sinkhorn normalization and the Frobenius inner product are differentiable w.r.t. M, gradients flow through $M = P^\top Z$ to the soft assignments P, enabling end-to-end training.

E.2 PIN decoder ablation baselines

To isolate the contribution of the decoder, we keep the PIN encoder unchanged and replace only the PIN decoder with two non-iterative alternatives. All variants take the same per-point representations $R^{(0)} \in \mathbb{R}^{N \times D}$ from the PIN encoder as input and output per-point cluster-assignment logits. For a fair comparison, we align model capacity by using the same embedding dimension, number of heads, and feedforward width, and by matching the total number of attention layers in the decoder (counting both self-attention and cross-attention).

E.2.1 Naïve (SA+MLP) decoder

The Naïve decoder predicts cluster assignments directly from point embeddings, without maintaining explicit cluster representations. We reuse the self-attention operator $\mathrm{SA}(\cdot)$ defined in Eq. (5) and stack $2L$ SA blocks (without positional encodings) to refine the point features:

$$R^{(l+1)} = \mathrm{SA}(R^{(l)}), \quad l = 0, \ldots, 2L-1, \qquad \tilde{R} = R^{(2L)} \in \mathbb{R}^{N \times d}.$$

We use $2L$ blocks to match the parameter budget of other ablations. We then apply a point-wise two-layer MLP head (hidden size $2d$) to produce logits over $K_{\max}$ clusters for every point:

$$\ell = \mathrm{MLP}(\tilde{R}) \in \mathbb{R}^{N \times K_{\max}}.$$

For a dataset with K clusters, we use the first K coordinates $\ell^{(K)} = (\ell_{:,1}, ..., \ell_{:,K}) \in \mathbb{R}^{N \times K}$ and mask the remaining coordinates as invalid. This baseline follows the SA+MLP intuition: stacked self-attention captures point–point interactions, while the MLP produces K-way assignment logits per point. Since clusters are only implicitly represented in the output layer, this baseline does not maintain cluster prototypes.

E.2.2 Non-iterative Transformer decoder

This baseline introduces explicit cluster representations but performs decoding without iterative refinement. We first compute set-aware point features by stacking L self-attention blocks (Eq. (5)) on the input set, without positional encodings:

$$R^{(l+1)} = \mathrm{SA}\Big(R^{(l)}\Big), \quad l = 0, \ldots, L-1, \qquad R^{(L)} \in \mathbb{R}^{N \times d}.$$

We then introduce $K_{\max}$ learnable cluster prototypes $C^{(0)} \in \mathbb{R}^{K_{\max} \times d}$ and update them for L layers using (i) prototype self-attention (Eq. (5)) and (ii) prototype-to-data cross-attention $\mathrm{CA}_{C \leftarrow R}(\cdot, \cdot)$ (Eq. (6)), both without positional encodings:

$$\tilde{C}^{(\ell)} = \mathrm{SA}\Big(C^{(\ell)}\Big), \qquad C^{(\ell+1)} = \mathrm{CA}_{C \leftarrow R}\Big(\tilde{C}^{(\ell)}, R^{(L)}\Big), \quad \ell = 0, \ldots, L-1.$$

Given the final prototypes $C^{(L)}$ and encoded point features $R^{(L)}$, we compute data-to-prototype assignment logits using the same projection, normalization, and similarity computation as the main model, producing $\ell \in \mathbb{R}^{N \times K_{\max}}$. For a dataset with K clusters, we select the first K coordinates to form $\ell^{(K)} \in \mathbb{R}^{N \times K}$ and mask the remaining coordinates as invalid. The rest of the pipeline follows the main experiments.

E.3 CIN ordinal loss ablation

CIN predicts the number of clusters K, which naturally has an ordinal structure. Our default CIN loss, however, treats $K \in \{2, \ldots, K_{\max}\}$ as a categorical label and optimizes a flat multi-class cross-entropy objective. To assess whether explicitly leveraging the ordering of K is beneficial, we replace this objective with an *order-aware ordinal logistic regression loss*, while keeping the PIN backbone, CIN fingerprint construction, and the overall training protocol unchanged. Concretely, for each candidate $K \in \{2, \ldots, K_{\max}\}$, we compute the same normalized Gram matrix $G^{(K)} = \frac{1}{N}(P^{(K)})^\top P^{(K)}$ and the same permutation-invariant fingerprint $g^{(K)} \in \mathbb{R}^{K(K+1)/2}$ as in App. B.2.2, and concatenate $\{g^{(K)}\}_{K=2}^{K_{\max}}$ into a fixed-length feature vector. Following our decoupled optimization design, we detach $\ell^{(K)}$ and hence $P^{(K)}$, $G^{(K)}$, and $g^{(K)}$, when forming the input features, so the

Table 19: **CIN loss ablation**. Median kMAE on synthetic and real datasets using different CIN losses.

CIN loss	Synthetic	Real
Order-aware	1.0	2.5
CE (Ours)	**0.0**	**1.0**

ordinal logistic regression loss only updates the head parameters and does not backpropagate into PIN.

The order-aware CIN head first maps the concatenated fingerprints through an MLP to produce a scalar score $\text{CIN}_{\text{Order-aware}}(X) \in \mathbb{R}$. We then introduce monotonic thresholds $\{b_K\}_{K=2}^{K_{\max}-1}$ parameterized by unconstrained variables $\{\delta_j\}_{j=1}^{K_{\max}-2}$ via

$$b_K = \sum_{j=1}^{K-1} \text{softplus}(\delta_j), \quad K = 2, \ldots, K_{\max} - 1,$$

which enforces $b_2 \leq b_3 \leq \cdots \leq b_{K_{\max}-1}$ automatically. With a positive scale s, the ordinal logits are defined as

$$\eta_K(X) = s(\text{CIN}_{\text{Order-aware}}(X) - b_K), \quad K = 2, \ldots, K_{\max} - 1,$$

where $\eta_K(X)$ is interpreted as the logit for the binary event $\Pr(K^\star > K \mid X)$. Given the ground-truth cardinality $K^\star$, we construct binary targets $\mathbb{I}[K^\star > K]$ and optimize the average binary cross-entropy across thresholds:

$$\mathcal{L}_{\text{ord}} = \frac{1}{K_{\max} - 2} \sum_{K=2}^{K_{\max}-1} \text{BCEWithLogits}(\eta_K(X), \mathbb{I}[K^\star > K]).$$

At inference time, we predict $\hat{K}$ by counting how many thresholds are exceeded:

$$\hat{K} = 2 + \sum_{K=2}^{K_{\max}-1} \mathbb{I}[\eta_K(X) \geq 0],$$

which yields $\hat{K} \in \{2, \ldots, K_{\max}\}$ and respects the natural ordering of K.

Tab. 19 reports the median kMAE of `TabClustPFN` pretrained with order-aware ordinal logistic regression loss. Overall, incorporating ordinal supervision does not yield improvements over the default cross-entropy training of CIN.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 explicitly state the contributions—a hybrid GMM+ZEUS pretraining prior, a decoupled PIN–CIN architecture for joint inference of $\boldsymbol{Z}$ and K in a single forward pass, the permutation-invariant SoftARI objective, and SOTA results on a 44-dataset real-world benchmark—all supported by the experiments in Sec. 4 and ablations in Sec. 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Sec. 4 and Sec. 6, including the operating regime ($N \leq 1000$, $D \leq 64$, $K \leq 10$) used in pretraining and evaluation, and the open directions of extending to noisy and missing-value settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results or theorems; the mathematical formulation is used to define the model and training objectives.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The prior-data generation, network architecture, loss functions, optimizer and learning-rate schedule, and evaluation protocol are fully described in Secs. 3–4 and Apps. B–C, and a public code repository is linked in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A public code repository is linked in the abstract (`https://github.com/Tianqi-Zhao/TabClustPFN`); all real-world benchmark datasets are drawn from public sources (OpenML-CC18, TabArena, AutoML, ZEUS, UCI) with their IDs/names listed in App. C.1, and the synthetic prior generators are described in App. B.1.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details, optimizer, learning-rate schedule, pretraining steps, model architecture, benchmark construction, preprocessing, and baseline settings are described in Secs. 3–4 and Apps. B–C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report median ARI/NMI ranks with IQRs (Tab. 1, Fig. 7). To assess significance across datasets, we conduct one-sided paired t-tests (Fig. 13) and Wilcoxon signed-rank tests (Fig. 14) on per-dataset paired ARI differences, and report critical-difference (CD) diagrams (Fig. 15) based on average ARI ranks; full details are in App. D.6.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 3 (Model pretraining) reports that pretraining is run on four RTX 5090 GPUs for 10,000 steps at a cost of approximately 92 GPU hours; per-iteration training time and per-dataset inference runtime are additionally reported in Tab. 5, Fig. 1, and Tab. 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics: it uses only synthetic data and publicly available, license-compliant tabular benchmarks, involves no human subjects or personally identifiable information, and preserves anonymity in submission.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational research on unsupervised tabular clustering with no direct path to high-stakes deployment; the potential positive impact is faster automated exploratory analysis, and we are not aware of specific negative societal impacts beyond those generic to clustering methods.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a tabular clustering PFN trained on synthetic priors and standard public benchmarks; it is not a generative model, language model, or scraped-data release, and poses no high risk of misuse that would require additional safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets are properly credited and used in accordance with their licenses: TabICL [28] (BSD 3-Clause), ZEUS [22] (Apache-2.0), iResNet [2] (MIT), MixSim [23] (GPL $\geq$ 2), `scikit-learn` baselines (BSD 3-Clause), and the public benchmarks OpenML-CC18 [4] (CC0/Public Domain per OpenML policy), TabArena [7] (Apache-2.0), the AutoML benchmark [10] (MIT), and the UCI clustering collection [24] (MIT).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the pretrained `TabClustPFN` model, the GMM and ZEUS prior-data generators through the public repository linked in the abstract, with accompanying documentation and reproducibility scripts.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so no IRB approval (or equivalent) is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core methodology; any LLM use was limited to writing/editing assistance and did not affect the scientific content or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.