BSTabDiff: Block-Subunit Diffusion Priors for High-Dimensional Tabular Data Generation

Al Zadid Sultan Bin Habib[1], **Md Younus Ahamed**[2], **Prashnna Kumar Gyawali**[3], **Gianfranco Doretto**[4], **Donald A. Adjeroh**[5]

[1,2,3,5]Lane Department of CSEE
West Virginia University, Morgantown, WV 26506, USA

[4]SCI Institute & Department of Biomedical Informatics
The University of Utah, Salt Lake City, UT 84112, USA

{{ah00069,ma00087}@mix,{prashna.gyawali,donald.adjeroh}@mail}.wvu.edu, doretto@utah.edu

https://github.com/zadid6pretam/BSTabDiff

Abstract

High-Dimensional Low-Sample Size (HDLSS) tabular domains (e.g., omics) are characterized by $n \ll m$, where n = number of samples, and m = number of features. Such domains often exhibit strong local correlation groups, sparse cross-group dependencies, heavy-tailed non-Gaussian marginals, heteroscedastic noise, and structured missingness, making direct density learning in $\mathbb{R}^m$ ill-conditioned since $n \ll m$. We propose BSTabDiff, a block-subunit generative framework that partitions the m observed features into M latent blocks ($M \ll m$) and generates each block via a shared low-dimensional subunit variable, concentrating global dependence learning in the compact block-latent space $\mathbb{R}^M$ while decoding to the full feature space with copula-driven dependence, flexible per-feature marginals, and explicit missingness mechanisms. BSTabDiff can induce a data-driven feature ordering and block construction before latent partitioning, allowing the model to better align with local dependency structure when such organization is beneficial. BSTabDiff supports modern deep priors on block latents, including diffusion and normalizing flows, enabling stable synthesis and controllable benchmark generation in the HDLSS regime. Empirically, BSTabDiff produces more realistic and more stable high-dimensional synthetic data when compared with unstructured tabular generators on HDLSS data.

1 Introduction

Synthetic data has become a practical lever for scaling learning systems when real-world data are scarce, siloed, expensive to curate, or too sensitive to share. In industry settings, synthetic generation is increasingly positioned as a way to bootstrap domain-specific datasets for training and evaluating modern AI pipelines (including agentic systems), helping mitigate data bottlenecks while enabling controlled coverage of rare or safety-critical cases (NVIDIA). At the same time, synthetic data is now also part of the training recipe for tabular foundation models: Prior-data Fitted Networks such as TabPFN variants are trained offline on large collections of synthetic datasets sampled from a prior to approximate Bayesian inference at test time (Hollmann et al., 2023, 2025; Grinsztajn et al., 2025). These trends motivate tabular generators that are not only of high-fidelity, but also scalable and controllable so they can serve as reliable engines for pretraining, simulation, augmentation, and benchmarking across domains.

However, many high-value scientific tabular domains live in the High-Dimensional Low-Sample Size (HDLSS) regime, where n samples are far fewer than m features ($n \ll m$) (Hall et al., 2005; Aoshima et al., 2018; Li et al., 2011). HDLSS data such as omics-like datasets further exhibit strong local correlation groups (modules) (Langfelder and Horvath, 2008), sparse cross-group dependence, heavy-tailed and non-Gaussian marginals, heteroscedasticity and overdispersion (Love et al., 2014; Robinson et al., 2010; Chen et al., 2014), and structured missingness mechanisms (Rubin,

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

1976; Little and Rubin, 2019). In this regime, directly learning dense dependence in $\mathbb{R}^m$ is often ill-conditioned. Meanwhile, sequence-style tabular generators that treat columns as tokens (e.g., LLM-based synthesis) can become computationally strained as m grows, since standard self-attention scales quadratically in sequence length (Vaswani et al., 2017; Borisov et al., 2023). These gaps leave a practical need for tabular generators that explicitly exploit HDLSS structure to achieve stable learning and efficient sampling at omics-scale dimensionalities.

Contributions. We introduce **BSTabDiff** (Block-Subunit Tabular Diffusion), a block-subunit generative framework tailored to HDLSS tabular data. Key novel elements include:

1) Block-subunit HDLSS generator: We propose a generative family that partitions the m observed features into M latent blocks ($M \ll m$), generating each block via a shared low-dimensional subunit variable while preserving feature-wise marginals and structured missingness. **2) Compact deep priors on block latents:** We concentrate global dependence learning in $\mathbb{R}^M$ by placing modern priors on block latents, including diffusion and normalizing flows, improving stability when $n \ll m$. **3) Structure-aware block construction:** We incorporate Graph-guided Ordering with Block Segmentation (GO-BS) to induce a data-driven feature ordering and block assignment before latent partitioning, allowing BSTabDiff to better align latent blocks with local dependency structure when such organization is beneficial. On the theory side, we prove that the GO-BS objective is NP-hard via reduction from minimum bisection, motivating the sparse approximate solver used in our implementation. **4) HDLSS-oriented modeling knobs and guarantees:** We provide a block-factorized learning signal and permutation-invariant identifiability (up to block relabeling), accommodating arbitrary observed feature order. **5) Empirical stability in high dimension:** We show improved realism and stability over unstructured tabular generators in HDLSS settings, enabling controllable benchmark generation and synthetic pretraining at high feature counts. **6) Energy-Carbon aware machine learning perspective:** In addition to synthetic utility, we introduce an energy-carbon-aware evaluation criterion for tabular generative models that jointly assesses TSTR (train on synthetic test on real) performance, energy consumption, and CO_2e emissions for training.

2 Related Work

A broad set of baselines for tabular data synthesis exists, including GAN/VAE-style generators, diffusion/score-based models, and LLM/foundation-model approaches, and we provide a detailed review in Appendix A1.

3 Methodology

Block-Subunit Generator: BSTabDiff introduces a tabular generative model that partitions features into blocks and assigns each block a shared latent "subunit" variable, enabling high-dimensional feature generation through low-dimensional block factors while preserving per-feature marginals and missingness patterns. This complements prior tabular synthesis frameworks (e.g., GAN and system-based generators) by explicitly targeting HDLSS structure (Xu et al., 2019; Patki et al., 2016), and is compatible with modern latent priors such as diffusion or flows (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Kotelnikov et al., 2023; Dinh et al., 2017; Papamakarios et al., 2021). Fig. 1 illustrates full generative architecture of BSTabDiff, including latent sampling, emission decoding, and permutation to observed space. Fig. 2 illustrates the subunit generation in BSTabDiff.

A Block-Subunit Generative Model for HDLSS Tabular Data

Motivation. Real HDLSS tabular datasets (e.g., omics) exhibit behaviors that differ qualitatively from classical asymptotics (Hall et al., 2005; Aoshima et al., 2018) and often show (i) strong local correlation groups (Langfelder and Horvath, 2008), (ii) sparse cross-group dependencies, (iii) heavy-tailed / non-Gaussian marginals, (iv) heteroscedastic noise (mean-variance coupling / overdispersion) (Love et al., 2014; Robinson et al., 2010; Chen et al., 2014), and often (v) structured missingness (Rubin, 1976; Little and Rubin, 2019). We formalize a generative family that captures these properties while remaining analyzable in the $n \ll m$ regime. Our model preserves the core intuition of block structure (namely, shared latent factor per group (Bartholomew et al., 2011)) but extends it beyond simple Gaussian mean-shifts to support realistic marginals, mixed data types, and cross-block coupling. We assume each feature is distinct.

Latent Block Structure with Observed Feature Permutation. Let $X \in \mathbb{R}^m$ be a sample with

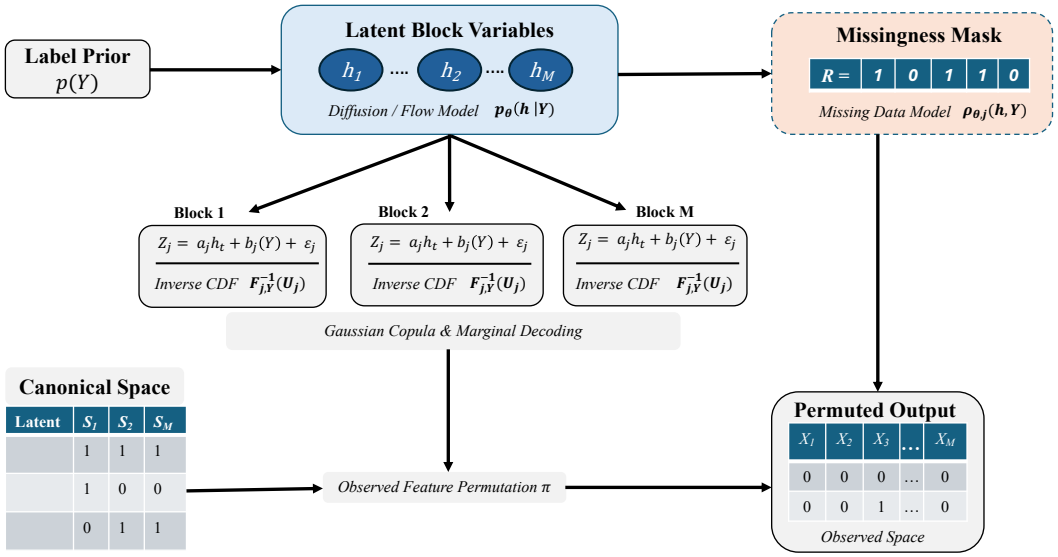


Figure 1: **Architecture of BSTabDiff.** The model samples a label, then draws low-dimensional block-latents $h_1, \ldots, h_M$ using a learned diffusion/flow prior. Each block governs a subset of features via copula-Gaussian decoding and inverse marginal CDFs to yield realistic marginals. A missingness mask is generated in parallel. The output is then restored to the original feature order, yielding high-dimensional tabular data with structured dependence, marginals, and missingness.

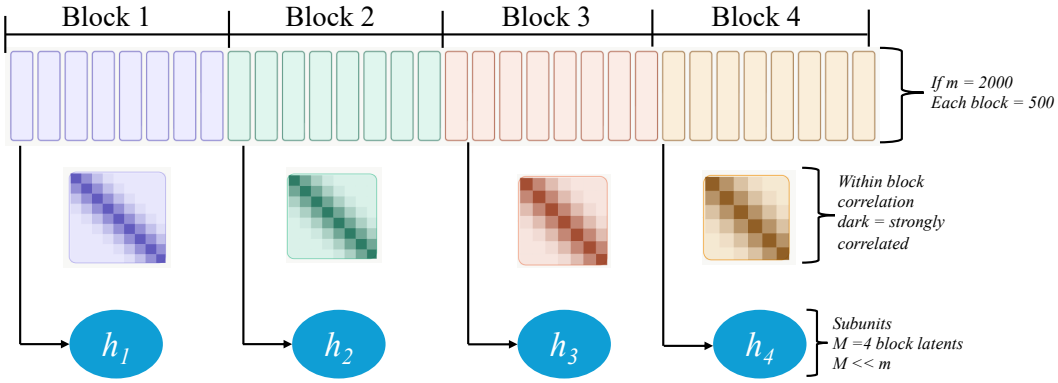


Figure 2: **Subunit generation in BSTabDiff.** BSTabDiff groups m HDLSS features into $M \ll m$ contiguous blocks, each governed by a shared latent subunit h_t. Features are decoded via Gaussian copulas with feature specific noise and learned inverse CDFs.

m features and optional label $Y \in \{1, \ldots, C\}$. Assume an (unobserved) partition of the canonical feature indices into M disjoint blocks $\{\mathcal{S}_t\}_{t=1}^{M}$ of sizes s_t (not necessarily equal), with $\sum_{t=1}^{M} s_t = m$. To decouple the canonical block index space from the observed feature order, we introduce a permutation $\pi \in \mathfrak{S}_m$ relating $\tilde{X}$ to X via $X = \tilde{X}_\pi$ (and similarly $R = \tilde{R}_\pi$), where $R \in \{0, 1\}^m$ is the binary observation mask ($R_j = 1$ observed, $R_j = 0$ missing/NA). In the GO-BS configuration, the feature permutation $\hat{\pi}$ and contiguous block boundaries $\hat{\mathbf{b}}$ are estimated from the training data before fitting the block-subunit generator. Following Appendix A2, the observed features and missingness indicators are reordered into canonical space as $X^\dagger = X_{\hat{\pi}^{-1}}$ and $R^\dagger = R_{\hat{\pi}^{-1}}$, where $\hat{\pi}$ maps original feature indices to their canonical ordered positions. The resulting contiguous blocks define feature subsets governed by individual block latents. This data-driven organization builds on the broader motivation for feature ordering and dependency-aware feature sequencing (Wang et al., 2025a; Habib et al., 2024, 2026c), while GO-BS additionally optimizes block segmentation for generative modeling. When feature ordering is disabled, $\hat{\pi} = \mathrm{id}$ and original feature order is retained. During generation, synthetic samples produced in canonical space are mapped back to original observed feature order using fixed learned permutation, rather than estimating or randomly sampling a new permutation.

Definition 3.1 (Block-subunit HDLSS generative model)**.** We fix blocks $\{\mathcal{S}_t\}_{t=1}^{M}$ in canonical index space. For each sample, generate:

1. **Label (optional):** $Y \sim p(Y)$.

2. **Block latents with cross-block dependence:** $h = (h_1, \ldots, h_M) \in \mathbb{R}^M$ from a label-conditional prior by Eq. 1 where p_θ captures cross-block dependence (e.g., sparse graphical prior, normalizing flow, diffusion, or mixture) (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Dinh et al., 2017; Papamakarios et al., 2021).

$$h \sim p_\theta(h \mid Y) \tag{1}$$

3. **Missingness mask (optional but realistic):** for each feature j in canonical space we get Eq. 2 reflecting structured missing-data mechanisms commonly modeled in statistical missingness theory (Rubin, 1976).

$$\tilde{R}_j \sim \text{Bernoulli}\big(\rho_{\theta,j}(h, Y)\big), \qquad \tilde{X}_j = \text{NA if } \tilde{R}_j = 0 \tag{2}$$

4. **Block-wise emissions (subunit measurements):** For each block t and each $j \in \mathcal{S}_t$ with $\tilde{R}_j = 1$, we draw an intermediate Gaussian copula variable (Nelsen, 2006) in Eq. 3 and map to $U_j = \Phi(Z_j) \in (0, 1)$, and define the observed canonical feature via an inverse marginal CDF by Eq. 4.

$$Z_j = a_j h_t + b_j(Y) + \xi_j, \qquad \xi_j \sim \mathcal{N}\big(0, \sigma_j^2(h_t, Y)\big) \tag{3}$$

$$U_j = \Phi(Z_j), \qquad \tilde{X}_j = F_{j,Y}^{-1}(U_j) \tag{4}$$

Here $F_{j,Y}$ is a (learned) marginal CDF allowing heavy tails, skew, and non-Gaussianity, matching common departures from Gaussian assumptions in some HDLSS data, such as omics datasets (Love et al., 2014; Robinson et al., 2010; Chen et al., 2014).

5. **Permutation to observed space:** We use the learned permutation $\hat{\pi}$ (identity if disabled) and output via Eq. 5.

$$X = \tilde{X}_{\hat{\pi}}, \qquad R = \tilde{R}_{\hat{\pi}} \tag{5}$$

Remarks. Eq. 3-4 yields a Gaussian copula dependence structure controlled by h_t, while $F_{j,Y}$ provides realistic feature-wise marginals (Nelsen, 2006; Sklar, 1959). Heteroscedasticity is captured via $\sigma_j^2(h_t, Y)$, and missingness is modeled via Eq. 2 (Love et al., 2014; Robinson et al., 2010; Chen et al., 2014; Rubin, 1976). This model strictly generalizes the simple additive Gaussian block factor model (shared factor + i.i.d. noise) (Bartholomew et al., 2011; Tipping and Bishop, 1999).

Deep Generative Parameterizations of the Block Prior. A key advantage in HDLSS is to learn global dependence at the block-latent level ($M \ll m$), where sample complexity is far more favorable than learning dense dependence directly in $\mathbb{R}^m$ (Hall et al., 2005; Aoshima et al., 2018).

Option A: Diffusion on h. We define a forward noising process $q(h_t \mid h_0)$ and train a conditional denoiser $\epsilon_\theta(\cdot)$ by Eq. 6. Then we sample $h_0 \sim p_\theta(h \mid Y)$ via reverse diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020) (see also score/SDE views (Song et al., 2021)) and decode to $\tilde{X}$ using Eq. 3-4. Diffusion models have also been adapted successfully to mixed-type tabular synthesis (Kotelnikov et al., 2023), and can be run in compact latent spaces (Rombach et al., 2022; Zhang et al., 2024).

$$\min_\theta \; \mathbb{E}_{(h_0, Y)} \, \mathbb{E}_{t,\epsilon}\Big[\big\|\epsilon - \epsilon_\theta(h_t, t, Y)\big\|_2^2\Big] \tag{6}$$

Option B: Normalizing flow on h. Let $h = f_\theta(\nu, Y)$ with base $\nu \sim \mathcal{N}(0, I)$. Then $p_\theta(h \mid Y)$ is tractable and can be trained by conditional maximum likelihood on inferred block latents (Dinh et al., 2017; Papamakarios et al., 2021).

Option C: Mixtures / graphical priors. For interpretability, $p_\theta(h \mid Y)$ can be a sparse Gaussian graphical model or a mixture of block states.

A Likelihood Factorization and a Block-Level Learning Signal. Let $\tilde{X}_{\mathcal{S}_t}$ denote the subvector in block t (canonical space). Under conditional independence of emissions given h (as in Definition 3.1), the joint likelihood factorizes across blocks as defined in Eq. 7.

$$p(\tilde{X}, \tilde{R} \mid h, Y) = \prod_{t=1}^{M} \prod_{j \in \mathcal{S}_t} p(\tilde{R}_j \mid h_t, Y) \; p(\tilde{X}_j \mid h_t, Y, \tilde{R}_j = 1) \tag{7}$$

Eq. 7 provides a direct learning signal for block structure: each h_t is responsible for explaining a coherent subset of correlated coordinates (Langfelder and Horvath, 2008), even if the observed coordinates are arbitrarily ordered (i.e., under an unknown permutation π). Here, emission factors are understood as observed-data likelihood contributions, taking value one when $\tilde{R}_j = 0$, while the missingness factors remain included for all features.

Permutation-Invariant Identifiability (Canonical Up to Block Permutations). Because the observed feature order is arbitrary, identifiability should be stated modulo permutation (Hyvärinen et al., 2009). We capture the standard notion: blocks are identifiable up to relabeling when they induce distinct dependence patterns.

Assumption 3.2 (Distinct block dependence)**.** For any two blocks $t \neq t'$, the pairwise dependence structure among coordinates in $\mathcal{S}_t$ differs from that in $\mathcal{S}_{t'}$ (e.g., different correlation spectra or different copula parameters). In addition, each feature is assumed to be unique, so coordinates are not exchangeable copies, even within a block. Cross-block dependencies are sparse in $p_\theta(h \mid Y)$.

Proposition 3.3 (Identifiability up to block permutation)**.** *Under Assumption 3.2 and with $a_j \neq 0$ for features that participate in block dependence, the partition $\{\mathcal{S}_t\}$ is identifiable from the population distribution of X up to permutation of block labels, even if the observed coordinates are permuted by an unknown π (Hyvärinen et al., 2009).*

Proof sketch. Because within-block dependence is induced through a shared scalar h_t (plus block-specific emissions), each block yields a characteristic dependence signature (e.g., rank-1 plus noise structure in copula Gaussian space). Distinctness implies there exists a clustering of coordinates that maximizes within-group dependence and minimizes cross-group dependence (e.g., as commonly done to identify correlated modules in high-dimensional biology (Langfelder and Horvath, 2008)). Unknown coordinate permutation changes feature order but not these dependence relations, hence the recovered partition matches the true one up to block relabeling. □

HDLSS Stability: Why Block Latents Are Learnable When $n \ll m$. The model is designed so the effective degrees of freedom scale with M rather than m, where M is the number of latent blocks (equivalently, the dimension of the block-latent vector $h \in \mathbb{R}^M$), aligning with HDLSS analyses that emphasize low-dimensional structure underlying high-dimensional observations (Hall et al., 2005; Aoshima et al., 2018).

Proposition 3.4 (Block-latent sample complexity advantage (informal))**.** *Assume $p_\theta(h \mid Y)$ has $O(M)$ to $O(M \log M)$ effective parameters (e.g., sparse couplings), and emissions $p(\tilde{X}_j \mid h_t, Y)$ share parameters within blocks. Then consistent estimation of the generative mechanism can be achieved with sample sizes scaling primarily with M (up to log factors and emission complexity), rather than with m (Hall et al., 2005; Aoshima et al., 2018). In contrast, unstructured generators that model dense dependence directly in $\mathbb{R}^m$ face parameter growth at least linear (often quadratic) in m, which is ill-conditioned in HDLSS regimes (Hall et al., 2005; Aoshima et al., 2018).*

Interpretation. Even when m is in the thousands and n is in the tens, block-level dependence can be learned because it compresses the global structure into $M \ll m$ latent variables and block-shared emissions (Hall et al., 2005; Aoshima et al., 2018).

SNR Scaling Within a Block (Connection to HDLSS Separability). To connect to the classical HDLSS characterization, we show how aggregation within a block boosts signal-to-noise ratio.

Lemma 3.5 (Within-block SNR scaling under mean-shift emissions)**.** *Consider a single block $\mathcal{S}$ of size s and a simplified continuous emission $\tilde{X}_j \mid Y = y \sim \mathcal{N}(\mu_y, \sigma^2)$ i.i.d. for $j \in \mathcal{S}$. Let $\bar{X}_\mathcal{S} = \frac{1}{s}\sum_{j\in\mathcal{S}} \tilde{X}_j$. Then we get Eq. 8.*

$$\mathrm{Var}(\bar{X}_\mathcal{S} \mid Y) = \sigma^2/s, \qquad \mathrm{SNR}(\bar{X}_\mathcal{S}) = \frac{(\mu_1 - \mu_0)^2}{\sigma^2/s} = s \cdot \frac{(\mu_1 - \mu_0)^2}{\sigma^2} \tag{8}$$

where μ_0 and μ_1 denote the class-conditional means for $Y = 0$ and $Y = 1$, respectively, i.e., $\mu_y \equiv \mathbb{E}[\tilde{X}_j \mid Y = y]$.

Takeaway. Blocks provide a natural mechanism for SNR amplification, which is critical when n is very small, as in HDLSS regimes (Hall et al., 2005; Aoshima et al., 2018). This intuition can be extended to non-Gaussian marginals and copula dependence via Eq. 4.

Synthetic HDLSS Dataset Generation as a Controlled Benchmark Suite. Definition 3.1 yields a family of synthetic HDLSS benchmarks with knobs controlling: (i) number of blocks M and size distribution $\{s_t\}$, (ii) cross-block sparsity/strength in $p_\theta(h \mid Y)$, (iii) marginal tail-heaviness via $F_{j,Y}$, (iv) heteroscedasticity via $\sigma_j^2(h_t, Y)$, (v) missingness mechanisms via $\rho_{\theta,j}(h, Y)$, and (vi) label-dependence via $b_j(Y)$ or label-conditional $p_\theta(h \mid Y)$. These knobs enable systematic evaluation of tabular generative modeling and synthetic pretraining in the HDLSS regime (Hall et al., 2005; Aoshima et al., 2018; Xu et al., 2019; Patki et al., 2016).

Generation Procedure. After training, synthetic generation follows a direct forward sampling pipeline that operates in the low-dimensional block-latent space and then decodes into the original m-dimensional feature space. The procedure produces (X, R, Y) and is summarized in Algorithm A10.1 in Appendix A10. Crucially, although the model uses only $M \ll m$ latent degrees of freedom through $h \in \mathbb{R}^M$, the output $X \in \mathbb{R}^m$ is not compressed: each feature j is generated explicitly via its block subunit and its learned marginal transform.

(1) Optional label sampling. If class-conditional generation is enabled, we either (i) sample Y from a specified class prior (uniform by default) or (ii) fix $Y = c$ to generate class-specific samples. For unconditional generation, Y is omitted and $h \sim p_\theta(h)$.

(2) Sample block latents from learned prior. We first sample the block latent vector in Eq. 9 where p_θ is implemented as a deep prior in $\mathbb{R}^M$. In our main instantiation, $p_\theta(h \mid Y)$ is a diffusion prior: sampling is performed by running the reverse diffusion chain from Gaussian noise to get h.

$$h = (h_1, \ldots, h_M) \sim p_\theta(h \mid Y) \tag{9}$$

This concentrates the model's global dependence learning in latent space, avoiding ill-conditioned high-dimensional density modeling in $\mathbb{R}^m$ in HDLSS settings (Hall et al., 2005; Aoshima et al., 2018). (For related tabular diffusion variants and latent-space diffusion instantiations, see (Kotelnikov et al., 2023; Zhang et al., 2024).)

(3) Sample missingness and emit features block-wise. Given h and (optionally) Y, we generate a binary observed-mask $\tilde{R} \in \{0, 1\}^m$ using the missingness model by Eq. 2 which can be unconditional or class-conditional and may vary across features, consistent with structured missing-data modeling (Rubin, 1976). If $\tilde{R}_j = 0$ we set $\tilde{X}_j = \texttt{NA}$; otherwise we emit $\tilde{X}_j$ using the block subunit corresponding to the block membership of feature j. For each continuous feature $j \in \mathcal{S}_t$ with $\tilde{R}_j = 1$, we first sample a Gaussian copula variable by Eq. 3 which maps it to a uniform variable $U_j = \Phi(Z_j)$, and then apply the learned inverse marginal CDF by Eq. 4.This yields (i) dependence driven by shared block subunit h_t in copula space and (ii) realistic, potentially heavy-tailed feature-wise marginals through $F_{j,Y}$, consistent with common departures from Gaussianity in omics-like measurements (Love et al., 2014; Robinson et al., 2010; Chen et al., 2014).

(4) Permutation to observed feature order. GO-BS estimates a dependency-aware permutation $\hat{\pi}$ and contiguous block boundaries from training data before fitting BSTabDiff. During generation, fixed learned permutation maps synthetic features & missingness masks from canonical space back to original feature order (Eq. 5). When GO-BS is disabled, $\hat{\pi} = \mathrm{id}$. Details are provided in App. A2.

Summary. Overall, generation follows the chain $Y \to h \to (\tilde{R}, \tilde{X}) \to (R, X)$, where the learned permutation $\hat{\pi}$ maps the canonical outputs back to the original feature order. The key HDLSS advantage is that global structure is learned and sampled in $\mathbb{R}^M$, while the full m-dimensional output is produced via block-conditioned emissions with learned marginals and missingness, yielding high-dimensional synthetic data (Hall et al., 2005; Aoshima et al., 2018).

Implementation note. In practice, once (θ, ϕ) are fitted, generation requires only sampling h from the prior and a single forward decode pass over features; no optimization is performed at generation time. This makes sampling efficient even when m is large (e.g., omics-scale) because all expensive dependence learning is confined to $M \ll m$.

Training procedure. We train the model by combining (i) a modern prior on block latents $p_\theta(h \mid Y)$ (diffusion or flow) and (ii) a block-factorized emission model $p_\phi(\tilde{X}, \tilde{R} \mid h, Y)$ (Eq. 7). In HDLSS, learning a high-capacity generator directly in $\mathbb{R}^m$ is often ill-conditioned; instead, we fit the global dependence in $\mathbb{R}^M$ and decode to $\mathbb{R}^m$ via block emissions.

Marginal estimation and Gaussianization. For each continuous feature j, we estimate empirical marginal CDFs $\hat{F}_{j,y}$ from observed training values, optionally conditioned on class $Y = y$, and

retain interpolated inverse CDFs $\hat{F}^{-1}_{j,y}$ for generation via Eq. 4. For latent inference, observed continuous values are independently rank-Gaussianized using pooled training samples across classes: $z_{ij} = \Phi^{-1}((r_{ij} - 0.5)/N_j)$, where r_{ij} is the observed rank and N_j is the number of observed values for feature j. Thus, forward Gaussianization is class-independent, whereas inverse marginal decoding can be class-conditional. All transformations are fitted within each training fold to avoid data leakage.

Latent inference. After GO-BS ordering and Gaussianization, block latents $h \in \mathbb{R}^M$ are inferred deterministically by averaging observed Gaussianized continuous features within each block and standardizing the resulting block factors. This lightweight factor-score construction (Kingma and Welling, 2014; Rezende et al., 2014) avoids training a separate inference encoder. Missing entries are excluded from block averages and emission fitting. The inferred latent vectors are subsequently used for block-conditioned emission estimation and joint prior learning. See Appendix A22 for further clarification on amortized inference.

Objective. We fit continuous emission parameters ϕ by least-squares regression on observed Gaussianized features, categorical emissions by cross-entropy, and observation probabilities from empirical missingness rates, optionally conditioned on class labels. These steps instantiate block-factorized model in Eq. 7 without directly maximizing its full joint likelihood. Compact prior $p_\theta(h \mid Y)$ is then trained on inferred block latents using conditional maximum likelihood for flows or denoising score matching for diffusion (Eq. 6). Algs. A10.1 and A10.2 in App. A10 detail generation and training.

Permutation estimation via GO-BS. BSTabDiff estimates an observed feature permutation $\hat{\pi}$ using GO-BS (Appendix A2), which orders dependent features contiguously to align the input space with the latent block structure. Given a weighted feature-affinity graph $G = (V, E, W)$, GO-BS estimates $\hat{\pi}$ and $M - 1$ boundaries $\hat{\mathbf{b}}$ by approximately minimizing Eq. 10.

$$\mathcal{J}_{\text{GO-BS}}(\pi, \mathbf{b}) = -\sum_{t=1}^{M} \sum_{\substack{i,j \in B_t(\pi,\mathbf{b}) \\ i<j}} W_{ij} + \lambda \sum_{1 \le t < t' \le M} \sum_{\substack{i \in B_t(\pi,\mathbf{b}) \\ j \in B_{t'}(\pi,\mathbf{b})}} W_{ij} + \gamma\, \Omega(\mathbf{b}) \tag{10}$$

where $B_t(\pi, \mathbf{b})$ is the t-th contiguous block. This objective rewards within-block affinity, penalizes cross-block leakage, and regularizes boundaries. Afterward, BSTabDiff trains in the GO-BS canonical order and maps generated samples back to the original feature order. Details, ablations, and pseudocode for BSTabDiff and GO-BS are provided in App. A2, A4, A5, A6, A7, and A10. We provide a (ρ, κ)-based GO-BS/GO-BS-FC selection rule with empirical validation in App. A12.1.

Micro-Macro dataset regime refinement. We extend the 5 dataset regimes based on $(n, m, \rho = m/n)$ by introducing $\kappa = nm$ as a complementary measure of data-matrix scale, partitioning each regime into Micro and Macro subsets using a regime-specific median threshold. Fig. A13.2 illustrates the 10 possible refined categories, with formal definitions provided in Appendix A13.2. Additional methodological clarifications concerning baseline selection, block construction, latent inference, marginal estimation, mixed-type handling, and practical configuration are provided in Appendix A22.

4 Experiments

In this section, we evaluate BSTabDiff against existing methods for HDLSS tabular generation.

Datasets. We evaluate HDLSS generation (Appendix A13) on eight publicly available real-world datasets (Table A1.2) from the repository (Li et al., 2018), previously used by Jiang et al. (2024) and Habib et al. (2026a); see Appendix A8 for extreme dimensional experiments. These datasets span diverse domains, with approximately 2K-20K+ numerical features, few samples, and no missing values in their processed versions. Several have also been studied in prior HDLSS research, including ProtoGate (Jiang et al., 2024), LLSPIN (Yang et al., 2022), and GOTabPFN (Habib et al., 2026a). We further evaluate nine additional datasets spanning diverse dimensionality regimes, including mixed-type features and heterogeneous missingness, as exemplified by TCGA-BRCA and Springleaf (Appendix A18). Overall, our experiments cover 18 distinct real-world datasets.

Baselines. Given the large and growing set of tabular generative models, we benchmark BSTabDiff against representative, widely used methods from each major family. We further restrict comparisons to baselines that provide publicly available implementations, and we exclude LLM-based generators because their computational overhead becomes prohibitive in high-dimensional settings. Specifically, we include SMOTE (Chawla et al., 2002) as a classical baseline; CTGAN (Xu et al., 2019), CTAB-GAN (Zhao et al., 2021), and CTAB-GAN+ (Zhao et al., 2024) as GAN-based methods; TVAE (Xu et al., 2019) as a VAE-based approach; and TabDDPM (Kotelnikov et al., 2023), TabDiff (Shi et al.,

Table 1: BSTabDiff compute, energy, and CO_2e for one run on Vulcan (no permutation), measured with CodeCarbon (Courty et al., 2026) and psutil (Rodola, 2026). See App. A9 and A17 for more.

Data	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)	Data	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)
COL	6.24	0.037	1.514	4.79e-4	4.29e-4	SMK	8.18	0.062	1.647	4.68e-4	4.20e-4
GLI	7.00	0.042	1.466	3.69e-4	3.31e-4	TOX	4.36	0.031	1.529	1.92e-4	1.72e-4
LNG	6.88	0.030	1.548	3.17e-4	2.84e-4	AML	6.57	0.036	1.532	1.31e-3	1.17e-3
PRS	5.28	0.209	2.519	2.52e-4	2.26e-4	ARC	9.34	0.030	1.697	4.30e-4	3.85e-4

Table 2: MLE-based TSTR synthesis comparison using Logistic Regression on 8 HDLSS datasets. Accuracy is reported as mean$_{\pm\text{std}}$ over evaluation folds. See App. A3 and A8 for more.

Model	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX
CTAB-GAN	$62.32_{\pm 5.56}$	$69.36_{\pm 3.56}$	$60.30_{\pm 6.68}$	$55.32_{\pm 6.68}$	$69.48_{\pm 6.24}$	$71.64_{\pm 5.47}$	$64.32_{\pm 6.86}$	$62.14_{\pm 6.45}$
CTAB-GAN+	$63.38_{\pm 7.32}$	$70.72_{\pm 4.86}$	$58.74_{\pm 5.56}$	$56.46_{\pm 5.56}$	$71.65_{\pm 5.46}$	$70.86_{\pm 6.36}$	$66.46_{\pm 4.56}$	$63.15_{\pm 5.46}$
ForestDiff	$74.65_{\pm 5.34}$	$76.17_{\pm 5.79}$	$72.34_{\pm 4.56}$	$62.14_{\pm 4.48}$	$80.81_{\pm 4.47}$	$76.12_{\pm 5.86}$	$70.15_{\pm 5.22}$	$76.12_{\pm 6.68}$
BinaryDiff	$72.16_{\pm 4.65}$	$73.76_{\pm 4.62}$	$70.71_{\pm 5.68}$	$61.13_{\pm 6.13}$	$78.67_{\pm 4.49}$	$73.71_{\pm 6.73}$	$69.12_{\pm 5.74}$	$71.69_{\pm 4.45}$
TabDiff	**79.72**$_{\pm 8.72}$	**88.64**$_{\pm 2.32}$	$76.48_{\pm 5.24}$	$68.32_{\pm 4.36}$	$92.68_{\pm 2.42}$	$86.42_{\pm 6.36}$	$76.72_{\pm 4.56}$	**86.32**$_{\pm 4.84}$
SMOTE	$78.71_{\pm 9.21}$	$87.03_{\pm 2.79}$	$76.47_{\pm 4.63}$	**68.44**$_{\pm 3.32}$	**93.22**$_{\pm 2.91}$	**87.16**$_{\pm 5.56}$	**79.08**$_{\pm 3.12}$	$84.32_{\pm 4.98}$
TVAE	$79.03_{\pm 6.35}$	$82.27_{\pm 3.15}$	**79.23**$_{\pm 4.81}$	$66.76_{\pm 4.19}$	$92.67_{\pm 3.43}$	$83.45_{\pm 7.81}$	$78.12_{\pm 6.76}$	$81.54_{\pm 5.23}$
CTGAN	$61.29_{\pm 7.56}$	$68.47_{\pm 3.89}$	$30.59_{\pm 7.26}$	$54.55_{\pm 5.98}$	$68.89_{\pm 6.21}$	$59.80_{\pm 8.76}$	$60.50_{\pm 5.34}$	$56.99_{\pm 5.07}$
TabDDPM	$69.35_{\pm 8.91}$	$68.47_{\pm 2.89}$	$65.88_{\pm 3.97}$	$54.01_{\pm 4.76}$	$73.61_{\pm 6.58}$	$69.17_{\pm 8.27}$	$65.12_{\pm 5.76}$	$59.77_{\pm 5.19}$
CtrTab	$43.38_{\pm 13.77}$	$58.60_{\pm 4.99}$	$62.35_{\pm 10.78}$	$55.05_{\pm 8.91}$	$41.62_{\pm 12.36}$	$35.30_{\pm 8.37}$	$47.00_{\pm 7.68}$	$22.83_{\pm 5.69}$
BSTabDiff[¶]	**83.21**$_{\pm 9.81}$	**94.48**$_{\pm 4.51}$	**84.94**$_{\pm 7.64}$	**69.39**$_{\pm 6.07}$	**95.56**$_{\pm 8.31}$	**88.40**$_{\pm 9.71}$	**85.30**$_{\pm 6.05}$	**89.36**$_{\pm 7.14}$
BSTabDiff[§]	**83.26**$_{\pm 12.40}$	**95.96**$_{\pm 2.85}$	**82.35**$_{\pm 8.96}$	**72.08**$_{\pm 5.58}$	**95.30**$_{\pm 7.45}$	**90.76**$_{\pm 9.25}$	**86.50**$_{\pm 6.24}$	**87.70**$_{\pm 6.20}$
Real (TRTR)	$86.13_{\pm 8.56}$	$96.54_{\pm 2.43}$	$89.88_{\pm 6.96}$	$72.34_{\pm 7.81}$	$98.04_{\pm 3.15}$	$91.76_{\pm 5.34}$	$86.70_{\pm 6.19}$	$94.38_{\pm 4.22}$

Notes. Gold, silver, and bronze cell colors denote the best, second-best, and third-best *synthetic* baselines in each dataset column, respectively. Real (TRTR) is shown only as an upper-bound reference and is not included in ranking. [§]: default configuration without tuning or permutation. [¶]: Optuna-tuned configuration without permutation. TSTR: train on synthetic, test on real; TRTR: train on real, test on real.

2025), BinaryDiffusion (Kinakh and Voloshynovskiy, 2024), ForestDiffusion (Jolicoeur-Martineau et al., 2024) as diffusion-style generators, and CtrTab (Li et al., 2026) for high-dimensionality.

Evaluation and Implementation. Our primary evaluation measure is Machine Learning Efficiency (MLE), following the standard protocol set by Kotelnikov et al. (2023); Kim et al. (2023); Lee et al. (2023); Zhang et al. (2024); Shi et al. (2025). MLE measures how well classifiers trained on synthetic data perform on a real test set (TSTR), with the upper bound given by training and testing on real data (TRTR). High-quality synthetic data should yield models that approach or sometimes exceed TRTR performance. We report ML efficiency using two protocols: (i) the common approach of averaging efficiency relative to a classical baseline (logistic regression) (Xu et al., 2019; Zhao et al., 2021; Kotelnikov et al., 2023), and (ii) an evaluation relative to strong modern tabular models CatBoost (Prokhorenkova et al., 2018), TANDEM (Naor and Lindenbaum, 2025), and TabPFN-2.5 (Grinsztajn et al., 2025) (for Colon), which represent competitive state-of-the-art baselines. We evaluate all classifiers using 5×5 cross-validation (5 repeats × 5 folds = 25 runs), following the HDLSS evaluation protocol established by Jiang et al. (2024) and Habib et al. (2026a). We also performed Optuna (Akiba et al., 2019)-based hyperparameter tuning with 2000 trials (see Appendix A16), following the tuning protocol of Kotelnikov et al. (2023) and Habib et al. (2026a). Experiments used compute resources: Vulcan (8×49 GB NVIDIA RTX A6000 GPUs, 2× Intel Xeon Gold 5320 CPUs, 503 GB RAM). Dataset and baseline source links are provided in Appendix A23.

Computational analysis. BSTabDiff trains a low-dimensional latent prior in $\mathbb{R}^M$ (with $M \ll m$) and decodes to m features via block-wise emissions. For a dataset with n samples, m features, M blocks, diffusion horizon T, and prior training epochs E, the dominant costs are (i) block-latent inference and emission fitting, which are linear in the observed entries and scale as $\mathcal{O}(nm)$ (e.g., Gaussianization/rank steps plus per-feature regression-like fits), and (ii) prior learning in latent space, which scales as $\mathcal{O}(E \cdot \min\{b, n\} \cdot T \cdot M \cdot H)$ for diffusion (or $\mathcal{O}(E \cdot \min\{b, n\} \cdot L \cdot M \cdot H)$ for a flow with L coupling layers), where b is batch size and H is the prior network width. Crucially, unlike generators that model dense dependence directly in $\mathbb{R}^m$, the expensive global dependence learning term depends on M rather than m, while the $\mathcal{O}(nm)$ emission-side work is a single pass over features and samples. This scaling is consistent with the empirical efficiency in Table 1: despite large m (up to 22,283 features in GLI-85 and 19,993 in SMK), training remains fast and memory-light, typically requiring under 10 seconds, below 0.1 GiB peak GPU memory, and about 1.5-1.7 GiB CPU RSS (Resident Set Size). This reflects the use of an M-dimensional latent prior and simple block-conditioned decoding. See statistical significance analysis in Appendix A15.

Table 3: COL downstream performance: BSTabDiff synthetic data vs. real data TRTR.

Data	+Acc	+AUC	∗Acc	∗AUC	◇Acc	◇AUC
BSTabDiff¶	$81.92_{\pm 10.56}$	**88.97**$_{\pm 10.33}$	$86.44_{\pm 9.80}$	$88.80_{\pm 8.11}$	**85.51**$_{\pm 10.16}$	$88.82_{\pm 8.97}$
BSTabDiff§	**81.97**$_{\pm 12.07}$	$87.85_{\pm 9.33}$	**86.74**$_{\pm 10.10}$	**89.25**$_{\pm 7.82}$	$85.46_{\pm 10.81}$	**89.22**$_{\pm 9.76}$
Real (TRTR)	$79.44_{\pm 11.10}$	$87.02_{\pm 9.25}$	$86.18_{\pm 8.68}$	$90.53_{\pm 8.64}$	$82.28_{\pm 7.86}$	$87.20_{\pm 9.68}$

Notes. Gold indicates the better mean result among the two BSTabDiff synthetic configurations for each metric. Real (TRTR) is shown separately as a reference. +: TANDEM; ∗: TabPFN-2.5; ◇: CatBoost. §: default, no tuning/permutation. ¶: Optuna-tuned, no permutation.

Table 4: **Extended TSTR AUC (%).** Results on nine additional datasets. Gold, silver, and bronze cells indicate the first, second, and third-highest distinct synthetic data results, respectively; ties share the same color. Bold values denote the best synthetic data result for each dataset. Standard deviations are shown as subscripts. TRTR is the real data reference and is excluded from ranking. The average is computed from the nine dataset-level means. See Tables A18.1 & A18.2 for TSTR acc. & fidelity.

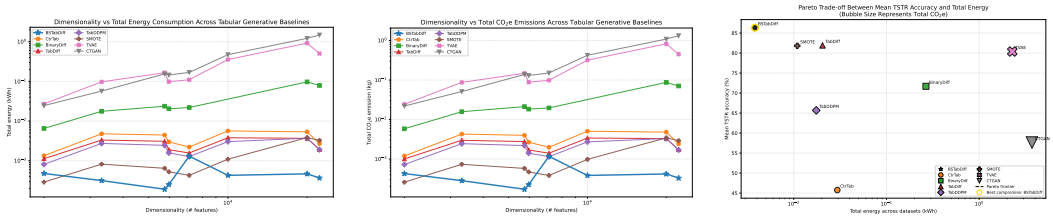
Model	TCGA BRCA ∗,†,‡	CIFAR-10 +	MiniBooNE #	Springleaf +,†,‡	ORLraws10P ∗	Adult #	BASEHOCK ◇	RELATHE ◇	PCMAC ◇	Avg.
BSTabDiff-no-p	**87.33**$_{\pm 2.88}$	**98.40**$_{\pm 0.09}$	$92.18_{\pm 1.44}$	$63.71_{\pm 1.37}$	**99.31**$_{\pm 0.72}$	$85.25_{\pm 1.08}$	**95.50**$_{\pm 3.12}$	$94.20_{\pm 1.51}$	$86.97_{\pm 4.15}$	**89.21**
BSTabDiff-BS	$87.25_{\pm 3.17}$	$98.39_{\pm 0.08}$	$91.74_{\pm 4.10}$	**63.79**$_{\pm 1.54}$	$99.30_{\pm 0.70}$	**85.59**$_{\pm 0.84}$	$94.61_{\pm 3.57}$	**94.25**$_{\pm 1.47}$	**87.03**$_{\pm 4.33}$	89.11
BSTabDiff-FC	$87.19_{\pm 3.34}$	**98.40**$_{\pm 0.10}$	$92.30_{\pm 1.39}$	$63.10_{\pm 1.54}$	$99.18_{\pm 0.83}$	$85.54_{\pm 1.12}$	$94.99_{\pm 3.35}$	$93.90_{\pm 1.37}$	$85.80_{\pm 5.36}$	88.93
BSTabDiff-d	$86.79_{\pm 3.73}$	$98.37_{\pm 0.10}$	**92.70**$_{\pm 0.85}$	$63.70_{\pm 1.31}$	$99.12_{\pm 0.91}$	$85.51_{\pm 0.91}$	$90.64_{\pm 1.89}$	$78.06_{\pm 2.90}$	$75.99_{\pm 3.55}$	85.65
TabDiff	$81.65_{\pm 5.54}$	$95.82_{\pm 0.83}$	$86.74_{\pm 2.02}$	$61.92_{\pm 3.33}$	$82.53_{\pm 7.98}$	$83.49_{\pm 1.64}$	$85.72_{\pm 5.15}$	$80.17_{\pm 6.79}$	$69.60_{\pm 7.96}$	80.85
CtrTab	$83.51_{\pm 4.20}$	$97.15_{\pm 0.31}$	$86.33_{\pm 1.90}$	$63.19_{\pm 1.51}$	$98.71_{\pm 0.54}$	$83.06_{\pm 1.87}$	$91.64_{\pm 2.65}$	$81.20_{\pm 2.75}$	$73.82_{\pm 3.52}$	84.29
TabDDPM	$32.26_{\pm 5.74}$	$51.39_{\pm 3.95}$	$82.53_{\pm 4.74}$	$50.45_{\pm 4.75}$	$49.44_{\pm 9.76}$	$78.75_{\pm 4.46}$	$51.34_{\pm 4.28}$	$52.01_{\pm 2.96}$	$50.81_{\pm 3.56}$	55.44
BinaryDiff	$52.49_{\pm 7.23}$	$51.16_{\pm 1.33}$	$58.91_{\pm 11.87}$	$51.39_{\pm 2.90}$	$82.89_{\pm 8.56}$	$80.44_{\pm 2.40}$	$58.40_{\pm 6.03}$	$55.63_{\pm 2.52}$	$57.54_{\pm 2.84}$	60.98
SMOTE	$86.39_{\pm 4.35}$	$97.74_{\pm 0.35}$	$90.56_{\pm 2.17}$	$62.83_{\pm 1.71}$	$98.91_{\pm 0.25}$	$85.58_{\pm 0.92}$	$88.40_{\pm 3.19}$	$80.12_{\pm 4.25}$	$75.87_{\pm 2.90}$	85.16
TRTR	$92.51_{\pm 2.52}$	$99.09_{\pm 0.09}$	$94.18_{\pm 0.19}$	$71.53_{\pm 0.74}$	$100.00_{\pm 0.00}$	$88.29_{\pm 0.53}$	$99.51_{\pm 0.17}$	$95.07_{\pm 1.34}$	$94.56_{\pm 1.03}$	92.75

Dataset notation: ∗HDLSS; +HDHSS; #LDHSS; ◇MixedRegime; †mixed continuous and categorical features; ‡feature-wise missing values. The three MixedRegime datasets are high-dimensional bag of words text representations.

Performance Comparison. Table 2 shows that BSTabDiff is the strongest synthetic data method under MLE-style Logistic Regression evaluation across 8 HDLSS datasets. Across its two configurations, BSTabDiff achieves the best synthetic accuracy on every dataset: the default setting performs best on COL, LNG, SMK, PRS, and ARC, while the tuned setting performs best on GLI, AML, and TOX. It also closely approaches the real data upper bound (TRTR) on several datasets, including LNG (95.96% vs. 96.54%), SMK (72.08% vs. 72.34%), ARC (86.50% vs. 86.70%), AML (95.56% vs. 98.04%), and PRS (90.76% vs. 91.76%). Table 3 further shows that, on Colon, BSTabDiff synthetic data remains competitive with real data training across TANDEM (Naor and Lindenbaum, 2025), TabPFN-2.5 (Grinsztajn et al., 2025), and CatBoost (Prokhorenkova et al., 2018), matching or slightly exceeding TRTR in several accuracy metrics while maintaining comparable AUC. Together, these results suggest that BSTabDiff preserves decision-relevant structure and produces high-utility synthetic samples that transfer across datasets and downstream evaluators in the HDLSS regime.

Generalization beyond the Original HDLSS Scope. We further evaluate BSTabDiff on nine additional datasets spanning HDLSS, HDHSS, LDHSS, and MixedRegime settings, including mixed-type omics and marketing data, image embeddings, and high-dimensional text representations. As shown in Table 4, at least one BSTabDiff variant achieves the highest synthetic data TSTR AUC on each dataset. BSTabDiff no-p obtains the best average AUC of 89.21%, outperforming the strongest non-BSTabDiff baseline (SMOTE, 85.16%) and approaching the real data TRTR reference (92.75%). These findings suggest that the block-subunit design generalizes beyond original HDLSS setting. Additional TSTR accuracy results and detailed analyses are deferred to Appendix A18.

Fidelity Diagnostics. BSTabDiff's fidelity diagnostics in Tables A11.1 and A11.2 (Appendix A11) show that synthetic samples generally preserve real marginals, dependence structure, and privacy separation without trivial memorization. Across both settings, per-feature KS / W1 distances remain relatively small on COL and AML, while LNG is consistently the most difficult case and GLI shows the largest scale-sensitive W1 and higher-order moment discrepancies, reflecting its extreme dimensionality and feature magnitude. Pearson/Spearman $|\Delta\mathrm{corr}|$ are typically moderate where defined, indicating substantial preservation of the correlation structure, and $|\Delta\mathrm{MI}(\mathrm{feature}, y)|$ remains reasonably controlled on COL, AML, and GLI but is weaker on LNG. NaN rates are negligible throughout, nearest neighbor privacy distances suggest limited sample copying, and in the default setting C2ST (Lopez-Paz and Oquab, 2017) is near chance for COL/AML/GLI but substantially higher on LNG, again identifying it as the hardest dataset. See Appendix A14 for t-SNE plots.



(a) Energy vs. dimensionality. (b) CO_2e vs. dimensionality. (c) TSTR-energy Pareto trade-off.

Figure 3: Energy-carbon-aware evaluation on HDLSS benchmarks. BSTabDiff achieves the best TSTR-energy-CO_2e trade-off and is the unique Pareto-efficient method. See Appendix A9 for details.

Energy-Carbon-Aware Utility Evaluation. Fig. 3 evaluates tabular generative models from an energy-carbon-aware perspective. While utility metrics such as TSTR accuracy are central, practical deployment also depends on computational cost, environmental footprint, and energy-reporting accountability (Lottick et al., 2019). Fig. 3a and 3b show that several deep tabular generators incur rapidly increasing energy and carbon dioxide equivalent (CO_2e) costs as dimensionality grows. In contrast, BSTabDiff remains consistently energy efficient across HDLSS datasets. Fig. 3c summarizes the trade-off between mean TSTR accuracy and total energy consumption, with bubble size indicating total CO_2e emissions. BSTabDiff attains the highest mean TSTR accuracy while also achieving the lowest total energy and CO_2e, yielding the best utility-sustainability trade-off among all evaluated methods. To reduce measurement noise, each run was executed as a single isolated process on a fixed GPU/node and monitored with `nvidia-smi/psutil` (Rodola, 2026); CodeCarbon (Courty et al., 2026) estimates local CPU/GPU/RAM energy only, so other nodes do not affect the estimate, while residual co-located process noise is treated as measurement uncertainty. See Appendix A9 for CodeCarbon (Courty et al., 2026) protocol details and dataset-wise energy/carbon results.

Ablation studies. Ablation results on Colon (TabPFN-2.5) show that BSTabDiff is broadly robust to reasonable hyperparameter changes, with TSTR AUC remaining in a tight range across compact variants (Fig. A12.1, Table A12.1 in Appendix A12). Disabling class-conditional marginals causes the clearest degradation in downstream utility, while switching to a flow prior and increasing the synthetic sample budget can slightly improve TSTR, with n_{syn}=500 achieving the best mean TSTR AUC. AUC efficiency (TSTR/TRTR) stays close to 1 across settings, indicating that synthetic-utility gains reflect real data performance rather than overfitting, and the scaling curves suggest only mild sensitivity to prior training epochs and the number of blocks M within the tested range. We provide a practical (ρ, κ)-based M initialization rule and a completed 5×5 CV sensitivity analysis of the block-latent dimension M in App. A19. A matched dense-factor ablation demonstrates the parameter efficiency and metric-dependent benefits of block-sparse generation (App. A20). A complete "one factor at a time" component sensitivity analysis and practical tuning guide are provided in App. A21.

Limitations and Impact Statement. BSTabDiff primarily targets HDLSS data ($n \ll m$), although our additional experiments demonstrate promising generalization across HDHSS, LDHSS, and mixed dimensionality regimes (Appendix A18). Its effectiveness depends partly on meaningful feature dependencies; when block structure is weak, GO-BS may introduce computational overhead with limited benefit. BSTabDiff supports synthetic data augmentation and benchmarking in data-scarce domains, with explicit energy and CO_2e reporting. Nevertheless, synthetic data should complement, not replace, real data validation and appropriate safeguards in high-stakes applications.

5 Conclusion

We introduced BSTabDiff, a block-subunit generative framework for HDLSS tabular synthesis that learns global dependence in a compact block-latent space ($M \ll m$) and decodes to high-dimensional observations through copula-driven, feature-wise structured emissions. By scaling with latent block dimension rather than ambient feature dimension, BSTabDiff enables stable generation when $n \ll m$. We further connect BSTabDiff with GO-BS, a graph-guided ordering and block segmentation procedure that aligns observed features into dependency-aware contiguous blocks. Empirically, BSTabDiff outperforms strong GAN/VAE/diffusion baselines, often approaches real data performance under classical and modern downstream classifiers, and preserves key marginal and dependence structure without trivial memorization. From a Green AI perspective, our energy-carbon-aware analysis shows that BSTabDiff achieves the best TSTR-energy-CO_2e trade-off among evaluated top methods. Overall, BSTabDiff provides an effective foundation for controllable HDLSS benchmark generation, data augmentation, and synthetic pretraining.

Acknowledgements

This work was supported in part by the US National Science Foundation under Awards #1920920, #2125872, #2223793, and #2318708. We sincerely thank the anonymous NeurIPS 2026 reviewers for their valuable feedback and constructive suggestions. We are especially grateful to the anonymous Area Chair for their thoughtful insights and guidance, which significantly contributed to strengthening this work. We also gratefully acknowledge Zaigham Abbas Randhawa, Pouya Iranmanesh, and Ghazaleh Mirzaee for their assistance with GPU clusters.

References

Shyam Agarwal and Mahasweta Chakraborti. The Hidden AI Race: Tracking Environmental Costs of Innovation. *arXiv preprint arXiv:2511.22781*, 2025. URL https://doi.org/10.48550/arXiv.2511.22781.

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating Inconsistent Information: Ranking and Clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008. URL https://doi.org/10.1145/1411509.1411513.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019. URL https://doi.org/10.1145/3292500.3330701.

Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. *arXiv preprint arXiv:2007.03051*, 2020. URL https://doi.org/10.48550/arXiv.2007.03051.

Makoto Aoshima, Dan Shen, Haipeng Shen, Kazuyoshi Yata, Yi-Hui Zhou, and James S Marron. A Survey of High Dimension Low Sample Size Asymptotics. *Australian & New Zealand Journal of Statistics*, 60(1):4–19, 2018. URL https://doi.org/10.1111/anzs.12212.

Jonathan E Atkins, Erik G Boman, and Bruce Hendrickson. A Spectral Algorithm for Seriation and the Consecutive Ones Problem. *SIAM Journal on Computing*, 28(1):297–310, 1998. URL https://doi.org/10.1137/S0097539795285771.

David J Bartholomew, Martin Knott, and Irini Moustaki. *Latent Variable Models and Factor Analysis: A Unified Approach*. John Wiley & Sons, 2011. URL https://doi.org/10.1002/9781119970583.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. URL https://archive.ics.uci.edu/dataset/2/adult.

Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A Review of Green Artificial Intelligence: Towards A More Sustainable Future. *Neurocomputing*, 599:128096, 2024. URL https://doi.org/10.1016/j.neucom.2024.128096.

Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language Models are Realistic Tabular Data Generators. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cEygmQNOeI.

Carlos M Carvalho, Jeffrey Chang, Joseph E Lucas, Joseph R Nevins, Quanli Wang, and Mike West. High-Dimensional Sparse Factor Modeling: Applications in Gene Expression Genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008. URL https://doi.org/10.1198/016214508000000869.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. URL https://www.jair.org/index.php/jair/article/view/10302.

Yunshun Chen, Aaron TL Lun, and Gordon K Smyth. Differential Expression Analysis of Complex RNA-Seq Experiments Using edgeR. *Statistical Analysis of Next Generation Sequencing Data*, pages 51–74, 2014. URL https://doi.org/10.1007/978-3-319-07212-8_3.

Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating Multi-Label Discrete Patient Records Using Generative Adversarial Networks. In *Machine Learning for Healthcare Conference*, pages 286–305. PMLR, 2017. URL https://proceedings.mlr.press/v68/choi17a.

Myung Jin Choi, Vincent YF Tan, Animashree Anandkumar, and Alan S Willsky. Learning Latent Tree Graphical Models. *The Journal of Machine Learning Research*, 12:1771–1812, 2011. URL https://www.jmlr.org/papers/volume12/choi11b/choi11b.pdf.

Nicos Christofides. Worst-Case Analysis of A New Heuristic for the Travelling Salesman Problem. In *Operations Research Forum*, volume 3, page 20. Springer, 2022. URL https://doi.org/10.1007/s43069-021-00101-z.

Giovanni Ciriello, Michael L Gatza, Andrew H Beck, Matthew D Wilkerson, Suhn K Rhie, Alessandro Pastore, Hailei Zhang, Michael McLellan, Christina Yau, Cyriac Kandoth, et al. Comprehensive Molecular Portraits of Invasive Lobular Breast Cancer. *Cell*, 163(2):506–519, 2015. URL https://doi.org/10.1016/j.cell.2015.09.033.

Benoît Courty, Victor Schmidt, Goyal-Kamal, inimaz, Marion Coutarel, Luis Blanche, Boris Feld, Jérémy Lecourt, Liam Connell, Amine Saboni, Patrick Lloret, SabAmine, supatomic, Mathilde Léval, cianc, Alexis Cruveiller, ouminasara, Franklin Zhao, Christian Bauer, Jerry Laruba Festus, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Alexandre Phiev, Edoardo Abati, Douglas Blank, and rosekelly6400. mlco2/codecarbon: v3.2.6, March 2026. URL https://doi.org/10.5281/zenodo.19334697.

Will Cukierski. CIFAR-10 - Object Recognition in Images. Kaggle Competition Dataset, 2013. URL https://www.kaggle.com/competitions/cifar-10. Accessed: July 25, 2026.

Ali Dashti, Ivan Komarov, and Roshan M D'Souza. Efficient Computation of k-Nearest Neighbour Graphs for Large High-Dimensional Data Sets on GPU Clusters. *PLOS One*, 8(9):e74113, 2013. URL https://doi.org/10.1371/journal.pone.0074113.

Richard Davis, Sanjay Chawla, and Philip Leong. Feature Graph Architectures. *arXiv preprint arXiv:1312.4209*, 2013. URL https://doi.org/10.48550/arXiv.1312.4209.

Sam Demharter and Valdemar Stentoft. BRCA Multi-Omics (TCGA). Kaggle Dataset, 2020. URL https://www.kaggle.com/datasets/samdemharter/brca-multiomics-tcga. Accessed: July 25, 2026.

Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006. URL https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.

Josep Díaz and George B Mertzios. Minimum Bisection Is NP-Hard on Unit Disk Graphs. *Information and Computation*, 256:83–92, 2017. URL https://doi.org/10.1016/j.ic.2017.04.010.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation Using Real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Wei Dong, Charikar Moses, and Kai Li. Efficient k-Nearest Neighbor Graph Construction for Generic Similarity Measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586, 2011. URL https://doi.org/10.1145/1963405.1963487.

Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 2002. URL https://doi.org/10.1109/4235.585892.

Gunter Dueck and Tobias Scheuer. Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, 90(1):161–175, 1990. URL https://doi.org/10.1016/0021-9991(90)90201-B.

Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context Optimization for Scalable Prior-Data Fitted Networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024. URL https://openreview.net/forum?id=F0fU3qhcIG.

CM Fiduccia and RM Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *19th Design Automation Conference*, pages 175–181. IEEE, 1982. URL https://doi.org/10.1145/62882.62910.

Fajwel Fogel, Alexandre d'Aspremont, and Milan Vojnovic. Spectral Ranking Using Seriation. *Journal of Machine Learning Research*, 17(88):1–45, 2016. URL https://www.jmlr.org/papers/v17/16-035.html.

Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. URL https://doi.org/10.1080/01621459.1937.10503522.

Michael R Garey, David S Johnson, and Larry Stockmeyer. Some Simplified NP-Complete Problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, pages 47–63, 1974. URL https://doi.org/10.1145/800119.803884.

gonza95c, Meghan O'Connell, Skyline, and Will Cukierski. Springleaf Marketing Response. Kaggle Competition Dataset, 2015. URL https://www.kaggle.com/competitions/springleaf-marketing-response. Accessed: July 25, 2026.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models. *arXiv preprint arXiv:2511.08667*, 2025. URL https://doi.org/10.48550/arXiv.2511.08667.

Al Zadid Sultan Bin Habib, Kesheng Wang, Mary-Anne Hartley, Gianfranco Doretto, and Donald A. Adjeroh. TabSeq: A Framework for Deep Learning on Tabular Data via Sequential Ordering. In *International Conference on Pattern Recognition*, pages 418–434. Springer, 2024. URL https://doi.org/10.1007/978-3-031-78128-5_27.

Al Zadid Sultan Bin Habib, Md Younus Ahamed, Prashnna Kumar Gyawali, Gianfranco Doretto, and Donald A. Adjeroh. GOTabPFN: From Feature Ordering to Compact Tokenization for Tabular Foundation Models on High-Dimensional Data. *Forty-Third International Conference on Machine Learning (ICML)*, 2026a. URL https://openreview.net/forum?id=fpqfV3lCIB.

Al Zadid Sultan Bin Habib, Md Younus Ahamed, Md Asif Bin Syed, Md Samiul Islam, Muntasir Tabasum, Tanpia Tasnim, and Md Ekramul Islam. AugTab: Learnable Feature Augmentation for Low-Dimensional Tabular Data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 413–431. Springer, 2026b. URL https://doi.org/10.1007/978-3-032-37670-1_24.

Al Zadid Sultan Bin Habib, Gianfranco Doretto, and Donald A. Adjeroh. DynaTab: Dynamic Feature Ordering as Neural Rewiring for High-Dimensional Tabular Data. In *Proceedings of the First Workshop on NeuroAI Multimodal Intelligence @ AAAI 2026*, volume 308 of *Proceedings of Machine Learning Research*, pages 27–57. PMLR, 2026c. URL https://proceedings.mlr.press/v308/habib26a.html.

Aric A Hagberg, Daniel A Schult, and Pieter J Swart. Exploring Network Structure, Dynamics, and Function Using NetworkX. In *Proceedings of the Python in Science Conference*, pages 11–15. SciPy, 2008. URL https://doi.org/10.25080/TCWV9851.

Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting Things in Order: An Introduction to the R Package Seriation. *Journal of Statistical Software*, 25:1–34, 2008. URL https://doi.org/10.18637/jss.v025.i03.

Peter Hall, James Stephen Marron, and Amnon Neeman. Geometric Representation of High Dimension, Low Sample Size Data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(3):427–444, 2005. URL https://doi.org/10.1111/j.1467-9868.2005.00510.x.

Karen Hao. Training a Single AI Model Can Emit as Much Carbon as Five Cars. *MIT Technology Review*, June 2019. URL https://www.technologyreview.com/2019/06/06/239031. Accessed: 2026-04-23.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Stefanie Hölbling, Gottfried Kirchengast, Christian Briese, and Hans Thüminger. Energy Use and Carbon Emissions in High-Performance Computing: A Case Study for Universities and Reduction Strategies. *Cleaner Environmental Systems*, page 100332, 2025. URL https://doi.org/10.1016/j.cesys.2025.100332.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate Predictions on Small Data with a Tabular Foundation Model. *Nature*, 637(8045):319–326, 2025. URL https://doi.org/10.1038/s41586-024-08328-6.

Harold Hotelling. Analysis of A Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24(6):417, 1933. URL https://psycnet.apa.org/doi/10.1037/h0071325.

Aapo Hyvärinen, Jarmo Hurri, and Patrik O. Hoyer. Independent component analysis. In *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pages 151–175. Springer, 2009. URL https://doi.org/10.1007/978-1-84882-491-1_7.

Xiangjian Jiang, Andrei Margeloiu, Nikola Simidjievski, and Mateja Jamnik. ProtoGate: Prototype-based Neural Networks with Global-to-local Feature Selection for Tabular Biomedical Data. In *International Conference on Machine Learning*, pages 21844–21878. PMLR, 2024. URL https://proceedings.mlr.press/v235/jiang24c.html.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and Imputing Tabular Data via Diffusion and Flow-based Gradient-Boosted Trees. In *International Conference on Artificial Intelligence and Statistics*, pages 1288–1296. PMLR, 2024. URL https://proceedings.mlr.press/v238/jolicoeur-martineau24a.html.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1zk9iRqF7.

Sungkyu Jung and JS Marron. PCA Consistency in High Dimension, Low Sample Size Context. *The Annals of Statistics*, 37(6B):4104–4130, 2009. URL https://doi.org/10.1214/09-AOS709.

Brian W Kernighan and Shen Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970. URL https://doi.org/10.1002/j.1538-7305.1970.tb01770.x.

Sunawar Khan, Naila Sammar Naz, Tehseen Mazhar, Muhammad Usman Tariq, Tariq Shahzad, Sghaier Guizani, and Habib Hamam. Green AI Techniques for Reducing Energy Consumption in AI Systems. *Array*, page 100652, 2025. URL https://doi.org/10.1016/j.array.2025.100652.

Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1mNssCWt_v.

Jinhee Kim, Taesung Kim, and Jaegul Choo. EPIC: Effective Prompting for Imbalanced-Class Data Synthesis in Tabular Data Classification via Large Language Models. *Advances in Neural Information Processing Systems*, 37:31504–31542, 2024. URL https://openreview.net/forum?id=d5cKDHCrFJ.

Vitaliy Kinakh and Slava Voloshynovskiy. Tabular Data Generation Using Binary Diffusion. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=lv234j8Hq7.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. URL https://openreview.net/forum?id=33X9fd2-9FyZd.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. URL https://doi.org/10.1126/science.220.4598.671.

Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions. *Genome Research*, 13(4):703–716, 2003. URL https://doi.org/10.1101/gr.648603.

Christopher Kolberg, Katharina Eggensperger, and Nico Pfeifer. TabPFN-Wide: Continued Pre-Training for Extreme Feature Counts. In *EurIPS 2025 Workshop: AI for Tabular Data*, 2025. URL https://openreview.net/forum?id=iT5ldbTl3q.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling Tabular Data with Diffusion Models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR, 2023. URL https://proceedings.mlr.press/v202/kotelnikov23a.html.

Marieke Lydia Kuijjer, Matthew George Tung, GuoCheng Yuan, John Quackenbush, and Kimberly Glass. Estimating Sample-Specific Regulatory Networks. *iScience*, 14:226–240, 2019. URL https://doi.org/10.1016/j.isci.2019.03.021.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning. *arXiv preprint arXiv:1910.09700*, 2019. URL https://doi.org/10.48550/arXiv.1910.09700.

Peter Langfelder and Steve Horvath. WGCNA: An R Package for Weighted Correlation Network Analysis. *BMC Bioinformatics*, 9(1):559, 2008. URL https://doi.org/10.1186/1471-2105-9-559.

Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green Algorithms: Quantifying the Carbon Footprint of Computation. *Advanced Science*, 8(12):2100707, 2021. URL https://doi.org/10.1002/advs.202100707.

Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, 13(2):129–170, 1999. URL https://doi.org/10.1023/A:1006529012972.

Chaejeong Lee, Jayoung Kim, and Noseong Park. CoDi: Co-Evolving Contrastive Diffusion Models for Mixed-Type Tabular Synthesis. In *International Conference on Machine Learning*, pages 18940–18956. PMLR, 2023. URL https://proceedings.mlr.press/v202/lee23i.html.

Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2018. URL https://doi.org/10.1145/3136625.

Jundong Li et al. Datasets (scikit-feature / Feature Selection @ ASU). Website, 2025. URL https://jundongl.github.io/scikit-feature/datasets. Accessed: 2025-07-25.

Shengqiao Li, E James Harner, and Donald A Adjeroh. Random KNN Feature Selection-A Fast and Stable Alternative to Random Forests. *BMC Bioinformatics*, 12(1):450, 2011. URL https://doi.org/10.1186/1471-2105-12-450.

Zuqing Li, Junhao Gan, and Jianzhong Qi. Towards Synthesizing High-Dimensional Tabular Data with Limited Samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 15207–15215, 2026. URL https://doi.org/10.1609/aaai.v40i18.38545.

Xiaofeng Lin, Chenheng Xu, Matthew Yang, and Guang Cheng. CTSyn: A Foundational Model for Cross Tabular Data Generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sh4FOyZRpv.

Roderick JA Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 2019. URL https://doi.org/10.1002/9781119482260.

Siyang Liu and Han-Jia Ye. TabPFN Unleashed: A Scalable and Effective Solution to Tabular Classification Problems. In *Forty-second International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/liu25cn.html.

Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: Generative Modelling for Tabular Data by Learning Relational Structure. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=fPVRcJqspu.

David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJkXfE5xx.

Kadan Lottick, Silvia Susai, Sorelle A Friedler, and Jonathan P Wilson. Energy Usage Reports: Environmental Awareness as Part of Algorithmic Accountability. *arXiv preprint arXiv:1911.08354*, 2019. URL https://doi.org/10.48550/arXiv.1911.08354.

Michael I Love, Wolfgang Huber, and Simon Anders. Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2. *Genome Biology*, 15(12):550, 2014. URL https://doi.org/10.1186/s13059-014-0550-8.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. URL https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

Diana Mahdessian, Anthony J Cesnik, Christian Gnann, Frida Danielsson, Lovisa Stenström, Muhammad Arif, Cheng Zhang, Trang Le, Fredric Johansson, Rutger Schutten, et al. Spatiotemporal Dissection of the Cell Cycle with Single-Cell Proteogenomics. *Nature*, 590(7847):649–654, 2021. URL https://doi.org/10.1038/s41586-021-03232-9.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When Do Neural Nets Outperform Boosted Trees on Tabular Data? *Advances in Neural Information Processing Systems*, 36:76336–76369, 2023. URL https://openreview.net/forum?id=CjVdXey4zT.

Nicolai Meinshausen and Peter Bühlmann. High-Dimensional Graphs and Variable Selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006. URL https://doi.org/10.1214/009053606000000281.

Erel Naor and Ofir Lindenbaum. Hybrid Autoencoders for Tabular Data: Leveraging Model-Based Augmentation in Low-Label Settings. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=l5N0P1UJKP.

NCBI. GEO Accession Viewer: GSE146773 ǁ National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146773, 2021. Accessed: 2026-04-22.

Roger B. Nelsen. *An Introduction to Copulas*. Springer, 2 edition, 2006. URL https://link.springer.com/content/pdf/10.1007/0-387-28678-0_5.pdf.

Peter Bjorn Nemenyi. *Distribution-Free Multiple Comparisons*. Princeton University, 1963. URL https://www.proquest.com/openview/c1f3e8829e8351e9c2a1c5e51778c6cf/1.

NVIDIA. Synthetic Data Generation for Agentic AI. https://www.nvidia.com/en-us/use-cases/synthetic-data-generation-for-agentic-ai/. Accessed 2026-02-07.

Cosmas I. Nwakanma and Anurag K. Srivastava. The Artificial Intelligence–Power Grid Nexus for Sustainability: AI–Power Grid Sustainability Nexus: Challenges and Path Forward. *IEEE Energy Sustainability Magazine*, 2(3):121–131, 2026. URL https://doi.org/10.1109/ESM.2026.3662669.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL https://www.jmlr.org/papers/v22/19-1028.html.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The Synthetic Data Vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE, 2016. URL https://doi.org/10.1109/DSAA.2016.49.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased Boosting with Categorical Features. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

Lennart Purucker, Andrej Tschalzev, Nick Erickson, Gioia Blayer, David Holzmüller, Alan Arazi, Alexander Pfefferle, Mustafa Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How General Are Tabular Foundation Models, Really? *arXiv preprint arXiv:2606.30410*, 2026. URL https://doi.org/10.48550/arXiv.2606.30410. Accepted at NeurIPS 2026 (Evaluations and Datasets Track).

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014. URL https://proceedings.mlr.press/v32/rezende14.

Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: A Bioconductor Package for Differential Expression Analysis of Digital Gene Expression Data. *Bioinformatics*, 26(1):139–140, 2010. URL https://doi.org/10.1093/bioinformatics/btp616.

Giampaolo Rodola. psutil: Cross-Platform Library for Process and System Monitoring in Python. https://github.com/giampaolo/psutil, 2026. Accessed: 2026-04-27.

Byron Roe. MiniBooNE Particle Identification. UCI Machine Learning Repository, 2005. URL https://archive.ics.uci.edu/dataset/199/miniboone+particle+identification.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution Image Synthesis with latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01042.

Donald B Rubin. Inference and Missing Data. *Biometrika*, 63(3):581–592, 1976. URL https://doi.org/10.1093/biomet/63.3.581.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. URL https://doi.org/10.1145/3381831.

Aditya Shankar, Yuandou Wang, Rihan Hai, and Lydia Y Chen. Harpoon: Generalised Manifold Guidance for Conditional Tabular Diffusion. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=G5g6tDg1ZE.

Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. URL https://doi.org/10.1109/34.868688.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. TabDiff: a Mixed-type Diffusion Model for Tabular Data Generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=swvURjrt8z.

M Sklar. Fonctions de Répartition à n Dimensions et Leurs Marges. In *Annales de l'ISUP*, volume 8, pages 229–231, 1959. URL https://hal.science/hal-04094463/document.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Hochan Son, Xiaofeng Lin, Jason Ni, and Guang Cheng. SYNTHONY: A Stress-Aware, Intent-Conditioned Agent for Deep Tabular Generative Models Selection. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2026. URL https://openreview.net/forum?id=cj4SNumWqf.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/pdf/ef0eadbe07115b0853e964f17aa09d811cd490f1.pdf.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019. URL https://doi.org/10.18653/v1/P19-1355.

Peishuo Sun, Ying Wu, Chaoyi Yin, Hongyang Jiang, Ying Xu, and Huiyan Sun. Molecular Subtyping of Cancer Based on Distinguishing Co-Expression Modules and Machine Learning. *Frontiers in Genetics*, 13:866005, 2022. URL https://doi.org/10.3389/fgene.2022.866005.

Muntasir Tabasum, Al Zadid Sultan Bin Habib, Tanpia Tasnim, Md Ekramul Islam, Md Younus Ahamed, and Md Asif Bin Syed. AquaAugmentor: A Novel Feature Augmentation Algorithm for Water Potability Prediction. In *2024 6th International Conference on Sustainable Technologies for Industry 5.0 (STI)*, pages 1–6. IEEE, 2024. URL https://doi.org/10.1109/STI64222.2024.10951101.

Abdulaziz Tabbakh, Lisan Al Amin, Mahbubul Islam, GM Iqbal Mahmud, Imranul Kabir Chowdhury, and Md Saddam Hossain Mukta. Towards Sustainable AI: A Comprehensive Framework for Green AI. *Discover Sustainability*, 5(1):408, 2024. URL https://doi.org/10.1007/s43621-024-00641-4.

Michael E Tipping and Christopher M Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999. URL https://doi.org/10.1111/1467-9868.00196.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Guanchu Wang, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Junpeng Wang, Xiaoting Li, Mingzhi Hu, Chia-Yuan Chang, and Xia Hu. Advancing Table Understanding of Large Language Models via Feature Re-ordering. *ACM SIGKDD Explorations Newsletter*, 27(1):112–123, 2025a. URL https://doi.org/10.1145/3748239.3748248.

Yuandou Wang, Filip Gunnarsson, and Rihan Hai. IMLP: An Energy-Efficient Continual Learning Method for Tabular Data Streams. *arXiv preprint arXiv:2510.04660*, 2025b. URL https://doi.org/10.48550/arXiv.2510.04660.

Yuxin Wang, Duanyu Feng, Yongfu Dai, Zhengyu Chen, Jimin Huang, Sophia Ananiadou, Qianqian Xie, and Hao Wang. HARMONIC: Harnessing LLMs for Tabular Data Synthesis and Privacy Protection. *Advances in Neural Information Processing Systems*, 37:100196–100212, 2024. URL https://openreview.net/forum?id=hHA9qrGZBe.

Lukas Wegmeth, Tobias Vente, Alan Said, and Joeran Beel. Green Recommender Systems: Understanding and Minimizing the Carbon Footprint of AI-Powered Personalization. *ACM Transactions on Recommender Systems*, 2025. URL https://doi.org/10.1145/3768626.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling Tabular Data Using Conditional GAN. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

Junchen Yang, Ofir Lindenbaum, and Yuval Kluger. Locally Sparse Neural Networks for Tabular Biomedical Data. In *International Conference on Machine Learning*, pages 25123–25153. PMLR, 2022. URL https://proceedings.mlr.press/v162/yang22i.html.

Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4Ay23yeuz0.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. CTAB-GAN: Effective Table Data Synthesizing. In *Asian Conference on Machine Learning*, pages 97–112. PMLR, 2021. URL https://proceedings.mlr.press/v157/zhao21a.

Zilong Zhao, Aditya Kunar, Robert Birke, Hiek Van der Scheer, and Lydia Y Chen. CTAB-GAN+: Enhancing Tabular Data Synthesis. *Frontiers in Big Data*, 6:1296508, 2024. URL https://doi.org/10.3389/fdata.2023.1296508.

A Appendix

This technical appendix supports our main paper *BSTabDiff: Block-Subunit Diffusion Priors for High-Dimensional Tabular Data Generation* (Accepted to the Fortieth Annual Conference on Neural Information Processing Systems - NeurIPS 2026). Specifically, it includes:

- Detailed Related Work in Appendix A1
- Permutation Estimation via Graph-guided Ordering with Block Segmentation (GO-BS) in Appendix A2
- BSTabDiff with GO-BS in Appendix A3
- Ablation on Sample Clustering vs. Feature Clustering in GO-BS in Appendix A4
- GO-BS Ordering Ablation against Classical Metaheuristics in Appendix A5
- Ordering-Solver Ablation on a Fixed Global Affinity Graph in Appendix A6
- Ablation of GO-BS in Extreme Dimensionality in Appendix A7
- BSTabDiff in Extreme Dimensionality in Appendix A8
- Energy Use, Carbon Emissions, and Green AI Considerations in Appendix A9
- Pseudocode in Appendix A10
- Additional Fidelity Diagnostics in Appendix A11
- Additional Ablation Studies in Appendix A12
- HDLSS Literature and Datasets Used in Appendix A13
- Representation Quality via t-SNE in Appendix A14
- Statistical Significance Analysis in Appendix A15
- BSTabDiff Hyperparameters in Appendix A16
- Additional Runtime and Resource Usage in Appendix A17
- Generalization across Diverse Data Regimes in Appendix A18
- Block-Latent Dimension Selection and Sensitivity Analysis in Appendix A19
- Necessity of the Block-Subunit Structure in Appendix A20
- Component Sensitivity Analysis and Practical Configuration Guide in Appendix A21
- Additional Methodological Notes and Clarifications in Appendix A22
- Dataset and Baseline Sources in Appendix A23

A1 Detailed Related Work

Prior work related to tabular generation includes both generative synthesis methods and feature augmentation approaches (e.g., AquaAugmentor (Tabasum et al., 2024) or AugTab (Habib et al., 2026b)). Here, we focus on generative models, since BSTabDiff is a generative framework for HDLSS tabular synthesis.

GAN/VAE-based tabular generators. Tabular synthesis has evolved from classical oversampling strategies such as SMOTE (Chawla et al., 2002) to deep generative models designed for mixed-type tables with complex dependencies. Early representative methods include CTGAN (Xu et al., 2019) (and its VAE counterpart, TVAE (Xu et al., 2019)), which introduced conditional training strategies to better handle skewed categorical variables; CTAB-GAN (Zhao et al., 2021), which improved mixed-type modeling and missing-value handling via redesigned encodings and conditional vectors; and CTAB-GAN+ (Zhao et al., 2024), which incorporated differential privacy mechanisms (e.g., DP-SGD) to strengthen privacy-utility trade-offs. Privacy-centric and domain-specific variants such as PATE-GAN (Jordon et al., 2018) and medGAN (Choi et al., 2017) further highlight the importance of controlled disclosure and domain constraints in synthetic data generation.

Diffusion and score-based tabular generators. More recently, diffusion/score-based methods have emerged as strong general-purpose approaches for tabular synthesis. TabDDPM (Kotelnikov et al., 2023) established diffusion as a competitive baseline for tabular generation, motivating subsequent refinements for mixed-type data and improved dependency modeling, including STaSy (Kim et al., 2023), CoDi (Lee et al., 2023), and TabDiff (Shi et al., 2025). Latent-space diffusion has also been explored to better manage heterogeneous feature types and reduce modeling difficulty, as in TabSyn (Zhang et al., 2024). Complementary hybrid directions combine diffusion/flow ideas with tree-based learners, such as ForestDiffusion (Jolicoeur-Martineau et al., 2024), while discretization-first strategies such as Binary Diffusion (Kinakh and Voloshynovskiy, 2024) provide alternative pipelines for handling tabular features. HARPOON (Shankar et al., 2026) is an inference-time, manifold-guided conditional tabular diffusion method designed mainly for tasks such as imputation and inequality constraints, but unlike our approach it does not explicitly model contiguous latent block structure, and thus does not address the sample-efficiency and structural compression benefits that are central to HDLSS generation.

LLM/foundation-model and structure-aware generators. Beyond diffusion, structure-aware models explicitly represent column relationships, exemplified by GOGGLE (Liu et al., 2023), which learns relational structure among features for improved synthesis. In parallel, LLM-based tabular synthesis treats rows as sequences and leverages pretrained language models: GReaT (Borisov et al., 2023) adapts autoregressive LMs with flexible conditioning, HARMONIC (Wang et al., 2024) emphasizes joint utility and privacy evaluation for LLM-based synthesizers, and EPIC (Kim et al., 2024) proposed effective prompting for imbalanced-class tabular synthesis. Finally, CTSyn (Lin et al., 2025) targets cross-table generalization using schema-conditioned latent diffusion over a shared representation space.

HDLSS-oriented and selection-based generative directions. Recent work has also explored complementary directions for wide tabular generation. TabPFN-Wide (Kolberg et al., 2025) develops a customized prior for synthetic HDLSS data generation and uses continued pre-training to adapt tabular foundation models to extreme feature counts. CtrTab (Li et al., 2026) proposes a control-conditioned diffusion model for HDLSS tabular synthesis, but most of its evaluation is on datasets with only moderate feature dimensionality and relatively large sample sizes, with Arcene being the main truly HDLSS case, so its empirical setting only partially matches the breadth implied by its title. SYNTHONY (Son et al., 2026) instead focuses on intent-conditioned selection, proposing a stress-aware framework for choosing among GAN, diffusion, and LLM-based tabular synthesizers under fidelity, privacy, and utility objectives.

Positioning. Unlike prior tabular generators that largely treat all features in a flat manner, operate directly in the ambient feature space, BSTabDiff is designed specifically for the HDLSS regime, where $n \ll m$ and feature dependencies are often structured into local correlation groups. Our key distinction is to introduce a block-subunit generative view that compresses global dependence learning into a low-dimensional block-latent space while retaining flexible feature-wise decoding through copula-based emissions, non-Gaussian marginals, and explicit missingness modeling. In this sense, BSTabDiff is not simply another diffusion-based tabular generator; rather, it contributes a structure-aware HDLSS generative framework that can use diffusion or flow priors within a block-factorized design, improving stability, and controllability for high-dimensional tabular synthesis. Table A1.1 and Fig. A1.1 position BSTabDiff relative to prior non-LLM tabular generators. BSTabDiff is evaluated on 18 real datasets against 10 distinct baselines, covering sample sizes from 62 to 130,064 and feature dimensions from 14 to 42,728. Although some methods evaluate on more datasets, their reported benchmarks predominantly cover lower-dimensional settings, whereas BSTabDiff additionally demonstrates applicability to extreme HDLSS data. Specifically, Fig. A1.1(b) highlights the initial eight HDLSS benchmarks and Cell Cycle, spanning 62-1,067 samples and 2,000-42,728 features, while Fig. A1.1(a) reflects the complete 18 dataset evaluation, including nine additional datasets across diverse dimensionality regimes. We exclude LLM-based or LLM-assisted proposed models from this comparison because tokenizing thousands of features can incur substantial computational costs, particularly under quadratic self-attention, making conventional LLM-based generation less suitable for direct extreme HDLSS synthesis. For CtrTab, we consider its full reported evaluation of nine datasets, comprising seven main benchmarks and two additional high-dimensional case studies, Stadyn and Arcene. Dataset details and original sources are provided in Appendix A23; see also Appendix A13 for related HDLSS literature. Table A1.2 summarizes eight HDLSS datasets spanning

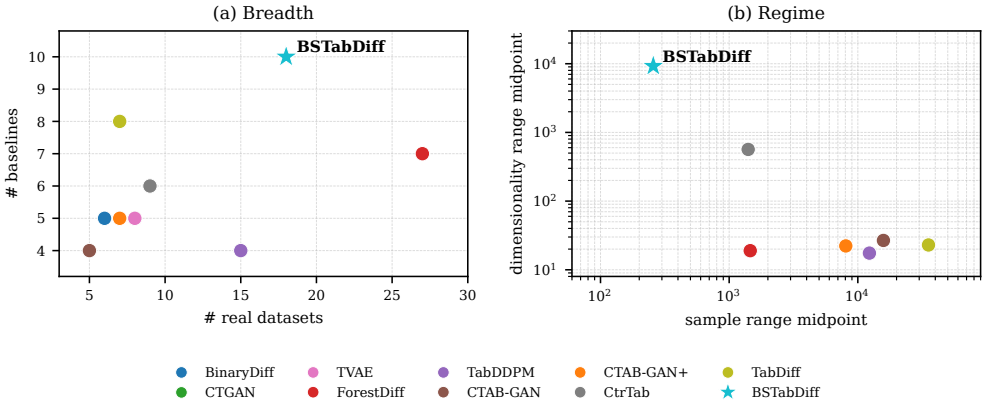


Figure A1.1: **Benchmark scope comparison of non-LLM tabular generative models.** (a) Evaluation breadth measured by the number of real datasets and distinct baseline methods; BSTabDiff covers 18 datasets and 10 baselines. (b) Benchmark regimes represented by the geometric midpoints of reported sample size and feature dimensionality ranges on logarithmic axes. For BSTabDiff, panel (b) specifically considers the eight original HDLSS benchmarks and the extreme dimensional Cell Cycle dataset ($n = 62$-1067, $m = 2000$-42728), highlighting its extreme HDLSS evaluation. The complete 18 dataset scope is summarized in Table A1.1.

Table A1.1: Benchmark scope of tabular synthetic data generation methods using real datasets only. LLM-based or LLM-assisted proposed models are excluded as rows, but LLM baselines are counted when used by non-LLM methods. "NR" indicates that the exact range was not reported. For CtrTab, we include its seven main benchmarks plus Stadyn and Arcene, following the full tested range in the original paper. For BSTabDiff, the evaluation includes eight initial HDLSS benchmarks, the extreme dimensional Cell Cycle dataset, and nine additional datasets spanning HDLSS, HDHSS, LDHSS, and mixed dimensionality regimes. The number of baselines denotes distinct comparison methods across the evaluation, not necessarily the number evaluated on every dataset. Dataset details are provided in Appendix A23, and the extreme dimensional evaluation is discussed in Appendix A8.

Model	# Real Datasets	# Baselines Compared	Sample Range	Dimensionality Range
BinaryDiff	6	5	NR	NR
CTGAN	8	5	NR	NR
TVAE	8	5	NR	NR
ForestDiff	27	7	[103 - 20640]	[4 - 90]
SMOTE	9	N/A	[937 - 443872]	NR
TabDDPM	15	4	[768 - 197080]	[6 - 51]
CTAB-GAN	5	4	[5000 - 50000]	[13 - 55]
CTAB-GAN+	7	5	[1300 - 50000]	[9 - 55]
CtrTab	9	6	[200 - 9873]	[32 - 10001]
TabDiff	7	8	[12330 - 101766]	[11 - 48]
BSTabDiff	**18**	**10**	**[62 - 130064]**	**[14 - 42728]**

62-203 samples, 2,000-22,283 features, and 2-5 classes, highlighting their extreme feature-to-sample ratios and varying class distributions. See other datasets in Table A23.3.

A2 Permutation Estimation via Graph-guided Ordering with Block Segmentation (GO-BS)

BSTabDiff estimates a dependency-aware feature permutation and contiguous block partition from the training data before fitting the block-subunit generator. GO-BS constructs a sparse weighted feature-affinity graph and seeks an ordering under which strongly dependent features become adjacent, followed by block segmentation that favors high within-block affinity and low cross-block leakage. Unlike a generic linear arrangement criterion, the GO-BS objective jointly considers feature ordering and block consistency. The estimated permutation and boundaries define the canonical feature organization for latent inference, emission fitting, and synthetic generation. This construction is related to classical seriation and graph partitioning methods (Atkins et al., 1998; Shi and Malik, 2000). Earlier work on feature graph architectures explored correlation-guided feature grouping and

Table A1.2: Summary of the HDLSS datasets used in our experiments (n=#samples, m=#features, C=#classes; distribution shown as class counts). See other datasets in Table A23.3.

Abbr	Name	n	m	C	Dist.	Abbr	Name	n	m	C	Dist.
COL	Colon	62	2000	2	[40, 22]	SMK	SMK	187	19993	2	[90, 97]
GLI	GLI-85	85	22283	2	[26, 59]	TOX	TOX171	171	5748	4	[45, 45, 39, 42]
LNG	Lung	203	3312	5	[139, 17, 21, 20, 6]	AML	ALLAML	72	7129	2	[47, 25]
PRS	Prostate	102	5966	2	[50, 52]	ARC	Arcene	200	10000	2	[112, 88]

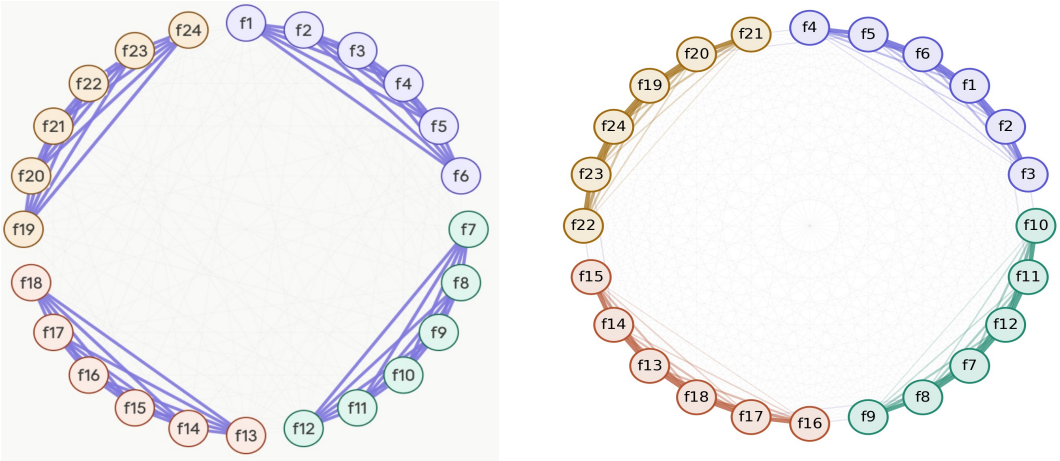


Figure A2.1: **Feature reordering in the feature-affinity graph.** Nodes $f_1, \ldots, f_{24}$ denote feature columns. In the arbitrary input order (left), related features need not be adjacent. We build a sparse feature-affinity graph by retaining the top-k strongest neighbors per feature; strong edges indicate within-group affinity and weak cross-group links are suppressed. After reordering (right), related features become contiguous, revealing dense local connectivity and reduced long range mixing.

permutation for hierarchical predictive learning (Davis et al., 2013); in contrast, GO-BS explicitly constructs a sparse feature-affinity graph and jointly estimates a dependency-aware ordering and contiguous block segmentation for generative modeling.

Feature graph. Let $X \in \mathbb{R}^{n \times m}$ be the training data matrix, where columns correspond to features and typically $n \ll m$. We construct a weighted undirected feature-affinity graph by Eq. 11.

$$G = (V, E, W), \qquad V = \{1, \ldots, m\} \tag{11}$$

where each vertex corresponds to one feature and $W_{ij} \geq 0$ measures affinity between features i and j (e.g., absolute correlation, cosine similarity, or a label-aware dependence score when labels are available). To scale to very high dimensions, we retain only the top-k strongest neighbors per feature and symmetrize the resulting graph. Thus the practical graph is sparse, with $|E| = O(mk)$, where k is a user-chosen sparsification parameter, typically treated as a small constant independent of m. To scale to very high dimensions, we sparsify the affinity graph by retaining only the top-k neighbors per feature, a standard strategy in k-nearest neighbor graph construction and approximation (Dong et al., 2011; Dashti et al., 2013). As shown in Fig. A2.1, reordering the feature-affinity graph places highly connected features adjacently, reducing long range mixing and revealing the underlying block structure.

Ordering and contiguous blocks. The ordering variable π plays the role of a seriation map, converting latent relational structure into an approximately contiguous one-dimensional arrangement (Atkins et al., 1998). A feature ordering is a bijection (Eq. 12).

$$\pi : V \to \{1, \ldots, m\} \tag{12}$$

or equivalently a permutation of the feature indices. Let the block count be M. Given an ordering π and a boundary vector is defined by Eq. 13.

$$\mathbf{b} = (b_1, \ldots, b_{M-1}), \qquad 1 \leq b_1 < \cdots < b_{M-1} < m \tag{13}$$

with the conventions $b_0 := 0$ and $b_M := m$, we define the induced contiguous ordered blocks

$$B_t(\pi, \mathbf{b}) = \{\, i \in V : b_{t-1} < \pi(i) \le b_t \,\} \qquad t = 1, \dots, M. \tag{14}$$

Hence GO-BS jointly estimates a permutation and a contiguous segmentation of that permutation. Our objective is inspired by classical graph partitioning principles, rewarding strong within-part connectivity while penalizing cross-part leakage, but is specialized here to contiguous blocks induced by a feature ordering (Kernighan and Lin, 1970; Shi and Malik, 2000).

Definition A2.1 (GO-BS block-consistency objective)**.** Given a weighted feature graph $G = (V, E, W)$, a target block count M, and nonnegative coefficients $\lambda, \gamma \ge 0$, define

$$\mathcal{J}_{\text{GO-BS}}(\pi, \mathbf{b}) = -\sum_{t=1}^{M} \sum_{\substack{i,j \in B_t(\pi,\mathbf{b}) \\ i<j}} W_{ij} + \lambda \sum_{1 \le t < t' \le M} \sum_{\substack{i \in B_t(\pi,\mathbf{b}) \\ j \in B_{t'}(\pi,\mathbf{b})}} W_{ij} + \gamma\, \Omega(\mathbf{b}) \tag{15}$$

where $\Omega(\mathbf{b})$ is a block-size regularizer. The GO-BS problem is

$$(\pi^*, \mathbf{b}^*) \in \arg\min_{\pi, \mathbf{b}} \mathcal{J}_{\text{GO-BS}}(\pi, \mathbf{b}) \tag{16}$$

The first term rewards within-block affinity, the second penalizes cross-block leakage, and the third regularizes the block sizes. A convenient choice is

$$\Omega(\mathbf{b}) = \sum_{t=1}^{M} \left(|B_t(\pi, \mathbf{b})| - \tau_t \right)^2 \tag{17}$$

where τ_t is a target size (e.g., $\tau_t = m/M$ for nearly balanced blocks). Other regularizers may be used to prevent degenerate blocks or enforce application-specific size priors.

Remark A2.2 (Equivalent cut form). Let

$$W_{\text{tot}} = \sum_{(i,j) \in E} W_{ij} \tag{18}$$

Since every edge either lies within one block or crosses between two distinct blocks, we have Eq. 19.

$$W_{\text{tot}} = \sum_{t=1}^{M} \sum_{\substack{i,j \in B_t(\pi,\mathbf{b}) \\ i<j}} W_{ij} + \sum_{1 \le t < t' \le M} \sum_{\substack{i \in B_t(\pi,\mathbf{b}) \\ j \in B_{t'}(\pi,\mathbf{b})}} W_{ij} \tag{19}$$

Therefore Eq. 15 can be rewritten, up to an additive constant, as

$$\mathcal{J}_{\text{GO-BS}}(\pi, \mathbf{b}) = -\,W_{\text{tot}} + (1 + \lambda) \sum_{1 \le t < t' \le M} \sum_{\substack{i \in B_t(\pi,\mathbf{b}) \\ j \in B_{t'}(\pi,\mathbf{b})}} W_{ij} + \gamma\, \Omega(\mathbf{b}) \tag{20}$$

Thus GO-BS is equivalently a contiguous balanced cut minimization problem.

Fig. A2.2 schematically illustrates the segmentation step after feature ordering, using a simplified contiguous feature layout for visual clarity.

Integration with BSTabDiff. Once GO-BS returns $(\hat{\pi}, \hat{\mathbf{b}})$, we reorder the observed data into canonical ordered space by

$$X^{\dagger} = X_{\hat{\pi}^{-1}} \tag{21}$$

The contiguous canonical blocks are then

$$\mathcal{S}_t = \{\, \hat{b}_{t-1} + 1, \dots, \hat{b}_t \,\}, \qquad t = 1, \dots, M \tag{22}$$

where $\hat{b}_0 = 0$ and $\hat{b}_M = m$. BSTabDiff is trained in this canonical ordered space using the block partition $\{\mathcal{S}_t\}_{t=1}^{M}$. At generation time, a synthetic canonical sample $\tilde{X}$ is mapped back to observed space by

$$X = \tilde{X}_{\hat{\pi}} \tag{23}$$

This turns the permutation mechanism into an explicit learned preprocessing stage that aligns observed coordinates with the block-subunit generator.

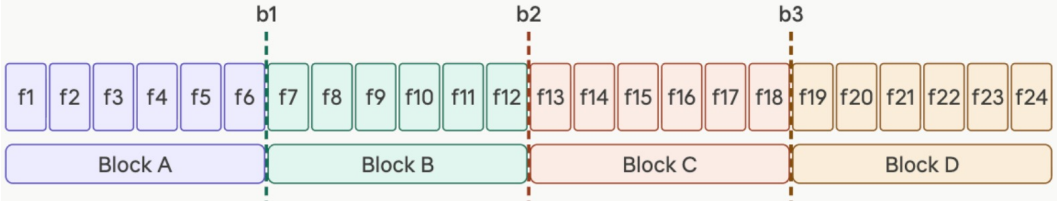


Figure A2.2: **Contiguous block segmentation in GO-BS.** Given an ordered feature sequence, GO-BS selects $M-1$ low-leakage boundaries to form M contiguous blocks, minimizing cross-block edge weight while preserving within-block affinity.

Decision version and complexity. We now show that GO-BS is computationally intractable in the worst case. We reduce from the minimum bisection problem, a classical NP-hard graph partitioning problem (Garey et al., 1974). Minimum bisection and related balanced partitioning problems remain central hard problems in graph partitioning theory (Garey et al., 1974; Díaz and Mertzios, 2017).

Definition A2.3 (Decision version of GO-BS)**.** Given a weighted graph $G = (V, E, W)$, an integer $M \geq 1$, a regularizer coefficient $\gamma \geq 0$, a leakage coefficient $\lambda \geq 0$, and a threshold $K \in \mathbb{R}$, the decision version of GO-BS asks whether there exists a permutation π and a valid boundary vector $\mathbf{b}$ such that Eq. 24 holds:

$$\mathcal{J}_{\text{GO-BS}}(\pi, \mathbf{b}) \leq K \tag{24}$$

Theorem A2.4 (NP-hardness of GO-BS)**.** *The GO-BS optimization problem in Eq. 16 is NP-hard. In fact, its decision version is NP-hard even in the restricted case of exactly two contiguous blocks of equal size.*

Proof. We reduce from the minimum bisection problem, which is NP-hard. An instance of minimum bisection consists of an undirected weighted graph $G = (V, E, W)$ with $|V| = m$ even, and a threshold K_{bis}. The question is whether there exists a partition by Eq. 25

$$V = S \cup \bar{S}, \qquad S \cap \bar{S} = \emptyset, \qquad |S| = |\bar{S}| = m/2 \tag{25}$$

such that the cut weight

$$\text{Cut}(S, \bar{S}) = \sum_{i \in S,\, j \in \bar{S}} W_{ij} \tag{26}$$

is at most K_{bis}. From this instance, construct a GO-BS instance on the same graph as follows Eq. 27.

$$M = 2, \qquad \lambda = 1, \qquad \gamma = 0 \tag{27}$$

and enforce the two block sizes to be exactly $m/2$ each by restricting the admissible boundaries to the unique value in Eq. 28.

$$b_1 = m/2 \tag{28}$$

Equivalently, one may keep $\gamma > 0$ and choose a hard balance regularizer that is zero only when $|B_1| = |B_2| = m/2$ and $+\infty$ otherwise; both formulations are equivalent for the reduction. Under this restriction, any permutation π induces exactly two contiguous blocks in Eq. 29

$$B_1(\pi) = \{\, i \in V : \pi(i) \leq m/2 \,\}, \qquad B_2(\pi) = \{\, i \in V : \pi(i) > m/2 \,\} \tag{29}$$

with $|B_1| = |B_2| = m/2$. Since $\gamma = 0$, the GO-BS objective reduces to

$$\mathcal{J}_{\text{GO-BS}}(\pi) = -\sum_{\substack{i,j \in B_1(\pi) \\ i<j}} W_{ij} - \sum_{\substack{i,j \in B_2(\pi) \\ i<j}} W_{ij} + \sum_{\substack{i \in B_1(\pi) \\ j \in B_2(\pi)}} W_{ij} \tag{30}$$

Using the decomposition of total weight in Eq. 31,

$$W_{\text{tot}} = \sum_{\substack{i,j\in B_1(\pi)\\ i<j}} W_{ij} + \sum_{\substack{i,j\in B_2(\pi)\\ i<j}} W_{ij} + \sum_{\substack{i\in B_1(\pi)\\ j\in B_2(\pi)}} W_{ij} \tag{31}$$

we obtain

$$\mathcal{J}_{\text{GO-BS}}(\pi) = -W_{\text{tot}} + 2 \sum_{\substack{i\in B_1(\pi)\\ j\in B_2(\pi)}} W_{ij} \tag{32}$$

Therefore minimizing $\mathcal{J}_{\text{GO-BS}}(\pi)$ is equivalent to minimizing the cut weight between the two induced equal-size blocks.

Now observe two directions.

(If) Suppose $(S, \bar{S})$ is a feasible minimum-bisection solution with $|S| = |\bar{S}| = m/2$ and $\text{Cut}(S, \bar{S}) \leq K_{\text{bis}}$. Construct any permutation π that lists all vertices of S in the first $m/2$ positions and all vertices of $\bar{S}$ in the last $m/2$ positions. Then $B_1(\pi) = S$ and $B_2(\pi) = \bar{S}$, so by Eq. 32, we get Eq. 33

$$\mathcal{J}_{\text{GO-BS}}(\pi) = -W_{\text{tot}} + 2\,\text{Cut}(S, \bar{S}) \leq -W_{\text{tot}} + 2K_{\text{bis}} \tag{33}$$

(Only if) Conversely, suppose there exists a GO-BS permutation π such that in Eq. 34

$$\mathcal{J}_{\text{GO-BS}}(\pi) \leq -W_{\text{tot}} + 2K_{\text{bis}} \tag{34}$$

By Eq. 32, the induced equal-size blocks satisfy Eq. 35.

$$\sum_{\substack{i\in B_1(\pi)\\ j\in B_2(\pi)}} W_{ij} \leq K_{\text{bis}} \tag{35}$$

hence $(B_1(\pi), B_2(\pi))$ is a feasible minimum bisection. Thus the minimum bisection instance is a YES-instance if and only if the constructed GO-BS instance is a YES-instance. The reduction is polynomial time. Since minimum bisection is NP-hard, the restricted GO-BS decision problem is NP-hard. Therefore the full GO-BS optimization problem is NP-hard as well. □

Remark A2.5. The hardness is driven by the combinatorial difficulty of deciding which features should belong to the same contiguous ordered block. The reduction does not rely on any generic linear arrangement term; it applies directly to the block-consistent objective itself.

A recovery-style sufficient condition. The NP-hardness result motivates approximate solvers. Still, the intended behavior of GO-BS can be formalized under a simple block-dominance condition. Proposition A2.7 formalizes the intended inductive bias behind GO-BS in the spirit of classical seriation and graph-partitioning methods (Atkins et al., 1998; Shi and Malik, 2000).

Assumption A2.6 (Ordered block-dominance)**.** There exists a ground-truth permutation $\pi^\dagger$ and a boundary vector $\mathbf{b}^\dagger$ inducing contiguous blocks in Eq. 36.

$$B_t^\dagger := B_t(\pi^\dagger, \mathbf{b}^\dagger), \qquad t = 1, \ldots, M \tag{36}$$

such that for some constants $a_{\text{in}}, a_{\text{out}} \geq 0$ with $a_{\text{in}} > a_{\text{out}}$, Eqs. 37 and 38 hold:

$$W_{ij} \geq a_{\text{in}} \quad \text{whenever } i, j \in B_t^\dagger \text{ for some } t \tag{37}$$

and

$$W_{ij} \leq a_{\text{out}} \quad \text{whenever } i \in B_t^\dagger,\ j \in B_{t'}^\dagger,\ t \neq t' \tag{38}$$

Proposition A2.7 (Exact recovery under strict block dominance)**.** *Assume $\gamma = 0$ and suppose Assumption A2.6 holds. If all admissible segmentations use the same block sizes as $\mathbf{b}^\dagger$, then every minimizer of Eq. 16 induces the same partition of V as $\{B_t^\dagger\}_{t=1}^M$, up to permutation of block labels and arbitrary within-block reorderings.*

Proof. Because all admissible segmentations have the same block sizes, the total number of within-block vertex pairs is fixed across feasible solutions, and so is the total number of cross-block pairs. For any feasible partition $\mathcal{P} = \{B_t\}_{t=1}^{M}$, denote by Eq. 39.

$$\mathrm{In}(\mathcal{P}) = \sum_{t=1}^{M} \sum_{\substack{i,j \in B_t \\ i<j}} W_{ij}, \qquad \mathrm{Out}(\mathcal{P}) = \sum_{1 \le t < t' \le M} \sum_{\substack{i \in B_t \\ j \in B_{t'}}} W_{ij} \tag{39}$$

Since $\gamma = 0$, minimizing Eq. 15 is equivalent to maximizing $\mathrm{In}(\mathcal{P})$, or equivalently minimizing $\mathrm{Out}(\mathcal{P})$.

Under Assumption A2.6, every pair placed correctly within its ground-truth block contributes at least a_{in}, while every pair split across distinct ground-truth blocks contributes at most a_{out}. Consider any feasible partition $\mathcal{P}$ that does not coincide with the ground-truth partition up to block relabeling. Then at least one pair of vertices that belong to the same ground-truth block is separated across different blocks of $\mathcal{P}$, or equivalently at least one pair from different ground-truth blocks is incorrectly grouped together. Because $a_{\mathrm{in}} > a_{\mathrm{out}}$, each such disagreement strictly decreases the within-block total relative to the ground-truth partition.

Therefore the ground-truth partition $\{B_t^{\dagger}\}_{t=1}^{M}$ uniquely maximizes $\mathrm{In}(\mathcal{P})$ among all feasible partitions, up to relabeling of the blocks. Since the objective depends only on block membership and not on the internal order of features inside each block, any within-block permutation yields the same optimum value. Hence every minimizer induces the same block partition up to block relabeling and within-block reorderings. □

Remark A2.8. Proposition A2.7 is intentionally simple: it formalizes the intended inductive bias of GO-BS, namely that if the affinity graph is approximately block-diagonal after some latent permutation, then the optimal segmentation recovers those blocks. This is directly consistent with BSTabDiff's identifiability discussion, where blocks are recoverable up to permutation when they induce distinct dependence patterns.

Sparse complexity. The exact optimization problem is NP-hard, so BSTabDiff uses a scalable sparse approximation. The key reason this is practical in HDLSS is that the sample dimension n is small even when the feature dimension m is large.

Proposition A2.9 (Sparse construction and refinement complexity)**.** *Assume each feature retains at most k neighbors in the affinity graph, so $|E| = O(mk)$. Then:*

1. *Constructing the sparse top-k graph from feature embeddings of dimension d costs $O(mdk)$ to $O(md \log m)$, depending on the neighbor-search routine;*
2. *Evaluating the block leakage term for fixed $(\pi, \mathbf{b})$ costs $O(mk)$;*
3. *One local boundary refinement sweep over all $M - 1$ boundaries with a constant-size search window costs $O(mk)$;*
4. *Hence P local refinement passes cost $O(Pmk)$ after graph construction.*

Proof sketch. Since each vertex has at most k retained neighbors after sparsification, the graph contains $O(mk)$ stored edges. Any objective term in Eq. 15 can therefore be evaluated by scanning only these edges once and checking whether the endpoints lie in the same block or in different blocks under the current segmentation; this yields $O(mk)$ time. For local boundary refinement, if each boundary is moved only within a constant-size neighborhood, then each candidate move changes the status of only edges incident to vertices near that boundary, and the aggregated work over a full sweep remains linear in the number of stored edges, i.e., $O(mk)$. Repeating for P passes gives $O(Pmk)$ after graph construction. □

Practical GO-BS solver. Exact optimization of Eq. 16 is infeasible for large m, so we adopt a sparse approximate solver. First, a sparse top-k feature-affinity graph is built on GPU when available. Second, a coarse dependency-aware ordering is obtained from the feature-affinity graph. Third, the ordered sequence is segmented into M contiguous blocks by minimizing the leakage-based boundary objective. Finally, local refinement updates both the ordering and the boundaries within short windows. Our practical solver combines sparse top-k graph construction, a graph-guided ordering

initialization, and local partition-refinement heuristics in the spirit of classical graph partitioning methods (Dong et al., 2011; Atkins et al., 1998; Kernighan and Lin, 1970; Fiduccia and Mattheyses, 1982).

GO-BS-FC variant. We also consider a feature-clustered variant, denoted GO-BS-FC, which preserves the same BSTabDiff interface and the same block-consistency goal as GO-BS, but changes the way the ordering problem is initialized. In standard GO-BS, samples are first clustered into regimes and each regime induces a feature-affinity graph over the original feature set, after which local feature orders are aggregated into a global permutation and segmented into contiguous blocks. In contrast, GO-BS-FC operates from the transposed view $X^\top \in \mathbb{R}^{m\times n}$: features themselves are treated as the clustering units, so the method first partitions the m feature vectors into M feature clusters and then forms blocks directly from these feature groups. Mathematically, standard GO-BS estimates both a permutation π and boundaries $\mathbf{b}$, inducing contiguous blocks $B_t(\pi, \mathbf{b})$, whereas GO-BS-FC estimates a feature-cluster assignment $c : V \to \{1, \dots, M\}$ and uses $B_t^{\mathrm{FC}} = \{i \in V : c(i) = t\}$, ordering features within each cluster to obtain a canonical sequence. Thus both variants seek high within-block affinity and low cross-block leakage, but GO-BS enforces contiguity through a learned global ordering, while GO-BS-FC obtains the block partition primarily through feature clustering. This makes GO-BS-FC especially attractive in extreme dimensional settings, because it avoids constructing and refining a dense global ordering over all m features while still producing the block partition required by BSTabDiff.

Use inside BSTabDiff. After GO-BS returns $(\hat{\pi}, \hat{\mathbf{b}})$, BSTabDiff applies Eqs. 21-23: the training data are reordered into canonical space, the induced contiguous segments become the latent blocks of the block-subunit generator, and synthetic samples are finally mapped back to the original observed coordinate indexing. This converts BSTabDiff's previously permutation mechanism into a learned, block-aware preprocessing stage tailored to the model's block-factorized generative structure.

A2.1 Assumption Diagnostics and Failure Modes

Meaning of sparse cross-block dependence. Assumption 3.2 introduces distinct within-block dependence structures and sparse cross-block interactions as structural conditions motivating the identifiability and sample complexity arguments of BSTabDiff. These conditions are not strict requirements for fitting the generative model. To clarify the meaning of sparsity, let $\mathcal{G}_h = (\{1, \dots, M\}, E_h)$ denote the conditional dependence graph of the M block latents, where an edge $(t, t') \in E_h$ indicates that h_t and $h_{t'}$ remain conditionally dependent given the other block latents and the optional label Y as Eq. 40.

$$(t, t') \in E_h \iff h_t \not\perp\!\!\!\perp h_{t'} \mid h_{-\{t,t'\}}, Y \tag{40}$$

Operationally, after discarding interactions below a small dependence threshold τ, we consider cross-block dependence effectively sparse when the resulting graph $E_h^{(\tau)}$ contains $O(M)$ or $O(M \log M)$ meaningful edges rather than $O(M^2)$. This distinction does not imply mutual independence among blocks: each block may interact strongly with a limited subset of other blocks, and the joint prior $p_\theta(h \mid Y)$ models these dependencies. The thresholded graph provides a practical approximation to the underlying conditional dependence structure rather than an exact conditional independence guarantee.

Practical verification of structural assumptions. The presence of recoverable block structure is primarily a property of the training data distribution rather than of a particular downstream classifier. We identify three complementary diagnostics for assessing the plausibility of the assumptions. **(1) Block coherence:** evaluate whether the GO-BS partition exhibits high within-block feature affinity and low normalized cross-block leakage, indicating that strongly dependent features are concentrated within the recovered blocks. **(2) Latent conditional dependence density:** estimate a thresholded conditional dependence graph from the inferred block latents h, using partial correlations or conditional mutual information, and examine whether its edge density is consistent with sparse interactions. Partial correlation provides a tractable diagnostic of linear conditional dependence, while conditional mutual information can capture more general relationships when estimation is reliable. **(3) Bootstrap stability:** repeatedly resample the training data, recover the feature partitions, and assess their consistency using partition-similarity measures that are invariant to block-label permutations. High block coherence, limited cross-block leakage, sparse estimated latent interactions,

and stable partitions provide practical evidence supporting the structural assumptions. However, these diagnostics are not formal tests of identifiability, and their reliability may be limited in extreme HDLSS settings.

Consequences of assumption violations. BSTabDiff does not fail automatically when cross-block dependencies are dense or the recovered feature blocks are weakly separated. The diffusion prior continues to learn a joint distribution $p_\theta(h \mid Y)$ and can represent dependencies among all block latents, including interactions that violate the effective sparsity assumption. Nevertheless, the conditions motivating Proposition 3.3 and the informal sample-complexity argument in Proposition 3.4 may no longer hold. In particular, block recovery can become less stable, and dependence fidelity may degrade when the available sample size is insufficient to estimate dense interactions among the block latents. Under such conditions, practitioners may reduce the cross-block leakage penalty, adjust the block budget M, disable GO-BS when ordering provides no validation benefit, or use $M = m$ to avoid block-level compression and approach a per-feature latent representation. These alternatives trade structural simplicity and computational efficiency against greater modeling flexibility.

Relationship to theoretical recovery conditions. The distinct dependence and sparsity assumptions motivate the identifiability and effective complexity arguments of BSTabDiff, but should not be interpreted as universally necessary conditions for successful generation. Appendix A20 provides a matched comparison with a dense latent-factor model, illustrating that block-subunit factorization can improve selected downstream utility and fidelity metrics while reducing decoder loading complexity. Separately, Assumption A2.6 and Proposition A2.7 provide stronger sufficient conditions for idealized GO-BS partition recovery under strict within-block dominance. The diagnostics described above offer practical guidance for assessing whether comparable structural organization is plausible in real datasets, without claiming that such measurements alone establish the theoretical recovery conditions.

A3 BSTabDiff with GO-BS

Downstream utility with permutation. We evaluate BSTabDiff with feature ordering using GO-BS and GO-BS-FC, with results summarized in Table A3.1. Compared with the no-permutation setting in Table 2, both ordering variants remain consistently competitive across all HDLSS datasets under TSTR evaluation. Across datasets, GO-BS-FC shows small but consistent gains in AUC. For example, on COL the AUC improves from 88.78% to 89.35% (+0.57%), on GLI from 89.79% to 89.96% (+0.17%), on SMK from 70.53% to 71.28% (+0.75%), on TOX from 89.76% to 90.01% (+0.25%), and on ARC from 73.37% to 77.14% (+3.77%), indicating improved discriminative structure in the generated data. The most pronounced improvement is observed on AML, where accuracy increases from 85.33% to 88.38% (+3.05%) and AUC from 89.88% to 94.69% (+4.81%). GO-BS remains competitive and occasionally slightly better in accuracy (e.g., PRS: 84.57% vs 83.94%, TOX: 89.35% vs 89.11%), but these differences are minor (typically $< 1\%$). On high-performing datasets such as LNG, both methods achieve near-identical results (e.g., AUC $\approx$ 99.55%-99.57%), suggesting limited headroom for improvement when performance is already saturated. Importantly, TRTR values remain unchanged and serve as the real data upper-bound reference. Overall, these results indicate that feature ordering does not drastically alter performance but provides consistent incremental gains, with GO-BS-FC offering a slightly more favorable tradeoff in downstream utility.

Runtime and resource usage. Table A3.2 shows that ordering remains practical across all datasets, with GPU memory generally below 1 GiB except for the largest feature spaces, GLI and SMK. GO-BS-FC is faster than GO-BS on most datasets, with especially large reductions on SMK (151.93s to 60.61s) and ARC (62.24s to 24.42s), while maintaining comparable GPU usage. GO-BS is only slightly faster on AML, and otherwise the feature-clustered variant is typically more efficient. Overall, GO-BS-FC provides the stronger runtime-structure tradeoff.

Fidelity diagnostics. Table A3.3 compares fidelity diagnostics for GO-BS and GO-BS-FC. Both variants produce numerically stable samples with zero NaN rates across all datasets. GO-BS more often achieves lower marginal and moment discrepancies, especially on TOX, AML, ARC, PRS, and LNG, while GO-BS-FC improves correlation preservation on several datasets, including COL, GLI, LNG, SMK, and TOX. GO-BS-FC also gives lower $|\Delta\mathrm{MI}|$ on LNG, PRS, SMK, and ARC, suggesting better label-feature alignment in some cases. Overall, ordering improves dataset quality in a metric-dependent way: GO-BS tends to favor marginal/moment fidelity, whereas GO-BS-FC often improves correlation or label-related structure while remaining competitive in downstream utility.

Table A3.1: Downstream utility of BSTabDiff with feature ordering under strict 5×5 TSTR evaluation using Logistic Regression. Results are mean$_{\pm \mathrm{std}}$ percentages; TRTR is the real data reference on the same folds. GO-BS denotes sample-clustered Graph-guided Ordering with Block Segmentation, and GO-BS-FC denotes its feature-clustered variant. Gold indicates the higher TSTR mean within each dataset; ties and TRTR references are unhighlighted.

Data	Ordering	TSTR Acc. (%) ↑	TSTR AUC (%) ↑	TRTR Acc. (%) ↑	TRTR AUC (%) ↑
COL	GO-BS	$82.87_{\pm 9.94}$	$88.78_{\pm 9.61}$	$86.13_{\pm 8.56}$	$87.45_{\pm 10.56}$
COL	GO-BS-FC	$\mathbf{83.21}_{\pm 10.09}$	$\mathbf{89.35}_{\pm 9.17}$	$86.13_{\pm 8.56}$	$87.45_{\pm 10.56}$
GLI	GO-BS	$84.94_{\pm 7.64}$	$89.79_{\pm 8.76}$	$89.88_{\pm 6.96}$	$93.42_{\pm 7.64}$
GLI	GO-BS-FC	$84.94_{\pm 7.64}$	$\mathbf{89.96}_{\pm 8.08}$	$89.88_{\pm 6.96}$	$93.42_{\pm 7.64}$
LNG	GO-BS	$94.20_{\pm 4.26}$	$\mathbf{99.57}_{\pm 0.40}$	$96.54_{\pm 2.43}$	$99.56_{\pm 0.44}$
LNG	GO-BS-FC	$\mathbf{94.58}_{\pm 4.28}$	$99.55_{\pm 0.42}$	$96.54_{\pm 2.43}$	$99.56_{\pm 0.44}$
PRS	GO-BS	$\mathbf{84.57}_{\pm 9.78}$	$81.57_{\pm 9.50}$	$91.76_{\pm 5.34}$	$94.91_{\pm 4.40}$
PRS	GO-BS-FC	$83.94_{\pm 8.46}$	$\mathbf{81.70}_{\pm 8.41}$	$91.76_{\pm 5.34}$	$94.91_{\pm 4.40}$
SMK	GO-BS	$69.07_{\pm 6.10}$	$70.53_{\pm 7.85}$	$72.34_{\pm 7.81}$	$79.13_{\pm 6.09}$
SMK	GO-BS-FC	$\mathbf{69.45}_{\pm 5.65}$	$\mathbf{71.28}_{\pm 7.22}$	$72.34_{\pm 7.81}$	$79.13_{\pm 6.09}$
TOX	GO-BS	$\mathbf{89.35}_{\pm 6.61}$	$89.76_{\pm 3.39}$	$94.38_{\pm 4.22}$	$99.50_{\pm 0.96}$
TOX	GO-BS-FC	$89.11_{\pm 7.32}$	$\mathbf{90.01}_{\pm 3.30}$	$94.38_{\pm 4.22}$	$99.50_{\pm 0.96}$
AML	GO-BS	$85.33_{\pm 7.82}$	$89.88_{\pm 9.77}$	$98.04_{\pm 3.15}$	$99.56_{\pm 0.87}$
AML	GO-BS-FC	$\mathbf{88.38}_{\pm 7.68}$	$\mathbf{94.69}_{\pm 6.89}$	$98.04_{\pm 3.15}$	$99.56_{\pm 0.87}$
ARC	GO-BS	$\mathbf{85.30}_{\pm 5.97}$	$73.37_{\pm 7.46}$	$86.70_{\pm 6.19}$	$91.91_{\pm 4.68}$
ARC	GO-BS-FC	$84.70_{\pm 6.96}$	$\mathbf{77.14}_{\pm 7.03}$	$86.70_{\pm 6.19}$	$91.91_{\pm 4.68}$

Table A3.2: Runtime, resource usage, and energy footprint of BSTabDiff with feature ordering. GPU memory denotes peak CUDA usage, CPU memory denotes end-of-run RSS, energy is in kWh, and CO_2e is in kg. GO-BS is the sample-clustered variant; GO-BS-FC is the feature-clustered variant. Gold indicates the lower value between the two ordering variants for each dataset and metric.

Data	Ordering	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)	Data	Ordering	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)
COL	GO-BS	11.30	**0.069**	**1.518**	5.77e-4	5.17e-4	**SMK**	GO-BS	151.93	**6.012**	2.264	7.19e-3	6.44e-3
COL	GO-BS-FC	**10.21**	0.087	1.825	**5.40e-4**	**4.84e-4**	**SMK**	GO-BS-FC	**60.61**	6.023	**1.806**	**3.73e-3**	**3.34e-3**
GLI	GO-BS	54.35	7.450	2.312	**3.11e-3**	**2.79e-3**	**TOX**	GO-BS	31.98	**0.523**	2.107	1.73e-3	1.55e-3
GLI	GO-BS-FC	**39.67**	**7.421**	**1.552**	6.50e-3	5.83e-3	**TOX**	GO-BS-FC	**27.50**	0.526	**1.802**	**8.13e-4**	**7.28e-4**
LNG	GO-BS	14.78	**0.177**	**1.505**	6.90e-4	6.18e-4	**AML**	GO-BS	**29.78**	**0.770**	**1.481**	**1.77e-3**	**1.58e-3**
LNG	GO-BS-FC	**10.88**	0.195	1.809	**5.71e-4**	**5.11e-4**	**AML**	GO-BS-FC	31.67	0.784	1.784	2.38e-3	2.13e-3
PRS	GO-BS	38.92	**0.557**	2.032	**2.18e-3**	**1.95e-3**	**ARC**	GO-BS	62.24	**1.534**	2.172	2.71e-3	2.43e-3
PRS	GO-BS-FC	**36.43**	0.563	**1.795**	5.29e-2	4.74e-2	**ARC**	GO-BS-FC	**24.42**	1.541	**1.831**	**2.03e-3**	**1.82e-3**

Effect of ordering on dataset quality. Comparing Table A3.3 with the no-permutation baseline in Table A11.2, feature ordering improves several important aspects of synthetic data quality. GO-BS strengthens marginal fidelity on multiple datasets, reducing KS/W1 or moment discrepancies on TOX, AML, GLI, and LNG, while GO-BS-FC improves structural preservation by lowering Pearson/Spearman correlation gaps on COL, SMK, LNG, and GLI. Ordering also improves label-related structure in several cases: GO-BS-FC lowers $|\Delta \mathrm{MI}|$ on LNG, PRS, SMK, and ARC, indicating better preservation of feature-label dependence. These gains occur while keeping NaN rates at zero and maintaining privacy distances comparable to the no-permutation setting, suggesting that ordering improves dataset quality without increasing numerical instability or sample-copying risk. Figure A3.1 summarizes the effect of feature ordering from complementary perspectives. GO-BS-FC improves TSTR AUC over GO-BS on most datasets, with especially visible gains on AML and ARC. GO-BS improves selected fidelity diagnostics relative to the no-permutation baseline, while GO-BS-FC reduces Pearson and Spearman correlation discrepancies on COL, indicating improved structural preservation. The metric-count summary further shows that ordering improves multiple fidelity criteria across datasets, supporting that feature ordering provides useful structural bias rather than only changing downstream scores.

Table A3.3: Fidelity diagnostics for BSTabDiff with feature ordering. Lower is better for KS, W1, correlation discrepancies, HOM, and $|\Delta\mathrm{MI}|$; higher privacy distance suggests less direct copying. Gold highlights the better result between GO-BS and GO-BS-FC within each dataset; ties are unhighlighted. KS = Kolmogorov-Smirnov, W1 = Wasserstein-1, HOM = higher-order moment discrepancy, MI = mutual information, and Priv. = nearest neighbor distance from synthetic to real samples.

Data	Ordering	KS mean	W1 mean	Pearson	Spearman	HOM mean	$\lvert\Delta\mathrm{MI}\rvert$	NaN	Priv. mean
COL	GO-BS	**0.1126**	**0.1760**	0.2188	0.2177	**1.42e-01**	**0.2213**	0	**5.01e1**
COL	GO-BS-FC	0.1168	0.1797	**0.1690**	**0.1684**	1.45e-01	0.2236	0	4.60e1
GLI	GO-BS	**0.0903**	**1.99e2**	0.2896	0.3136	**1.49e2**	**0.2116**	0	**4.81e5**
GLI	GO-BS-FC	0.0907	2.00e2	**0.2894**	**0.3132**	1.50e2	0.2119	0	4.79e5
LNG	GO-BS	**0.1987**	**0.1108**	0.1701	0.1774	**7.66e-02**	0.5723	0	1.73e1
LNG	GO-BS-FC	0.2005	0.1124	**0.1656**	**0.1746**	7.80e-02	**0.5676**	0	**1.76e1**
PRS	GO-BS	**0.1688**	**0.0290**	**0.2817**	**0.2324**	**2.01e-02**	0.1505	0	2.16e1
PRS	GO-BS-FC	0.1756	0.0326	0.2924	0.2508	2.43e-02	**0.1385**	0	**2.21e1**
SMK	GO-BS	0.1579	0.1699	0.3254	0.3088	8.76e-02	0.1728	0	**8.74e1**
SMK	GO-BS-FC	**0.1532**	**0.1531**	**0.3117**	**0.2972**	**4.84e-02**	**0.1653**	0	8.30e1
TOX	GO-BS	**0.0927**	**3.63e1**	0.1501	0.1493	**1.41e1**	**0.3031**	0	**3.71e4**
TOX	GO-BS-FC	0.1008	5.52e1	**0.1497**	**0.1492**	1.85e1	0.3086	0	3.36e4
AML	GO-BS	**0.0914**	**0.1402**	**0.1555**	**0.1592**	**1.02e-01**	**0.2145**	0	**9.35e1**
AML	GO-BS-FC	0.0955	0.1454	0.1563	0.1623	1.04e-01	0.2207	0	9.19e1
ARC	GO-BS	**0.1931**	**6.5563**	**0.1659**	**0.1578**	**5.1824**	0.0858	0	9.39e3
ARC	GO-BS-FC	0.2040	10.5546	0.1763	0.1642	8.2353	**0.0798**	0	**9.60e3**

A4 Ablation on Sample Clustering vs. Feature Clustering in GO-BS

We study a design choice in GO-BS: whether the graph-guided ordering pipeline should begin by clustering samples or by clustering features. Our main method uses sample clustering, from which we construct cluster-specific feature-affinity graphs, derive local feature orderings, and then aggregate them into a global feature order before contiguous block segmentation. Here, we compare this design against a feature-clustering alternative and against a simpler no-clustering baseline.

Motivation. The goal of GO-BS is to estimate a data-driven permutation under which strongly dependent features become contiguous and can be segmented into coherent blocks for BSTabDiff. This is closely related to seriation, i.e., finding a suitable linear order for a set of objects (Hahsler et al., 2008), such that more similar items are placed closer together in the recovered order (Fogel et al., 2016). A natural question is whether these blocks should be inferred from:

1. **sample-local feature structure**, obtained by first partitioning samples into local regimes and then constructing a feature graph within each regime; or
2. **global feature structure**, obtained by directly clustering features in the transposed data matrix.

GO-BS adopts the first strategy. The intuition is that in HDLSS data, especially in biological and biomedical settings, feature-dependence patterns may vary across subpopulations, phenotypes, or local sample regimes. This is consistent with prior work on heterogeneous biomedical tabular data, which emphasizes "the subset of most relevant features for each sample" (Yang et al., 2022), as well as work on sample-specific regulatory networks and subtype-specific co-expression modules (Kuijjer et al., 2019; Sun et al., 2022). A single global feature view may therefore blur regime-specific structure. Related work has explicitly contrasted sample-specific networks with aggregate ones (Kuijjer et al., 2019), and has shown that different cancer subtypes can exhibit distinct co-expression modules rather than one universal global module (Sun et al., 2022). By contrast, sample clustering allows GO-BS to recover multiple local feature graphs, each capturing dependence patterns specific to a subset of samples. This design is motivated by prior evidence that network structure can be sample-specific (Kuijjer et al., 2019) or subtype-specific (Sun et al., 2022), rather than being adequately summarized by a single global dependency graph.

Compared Variants. We compare the following variants.

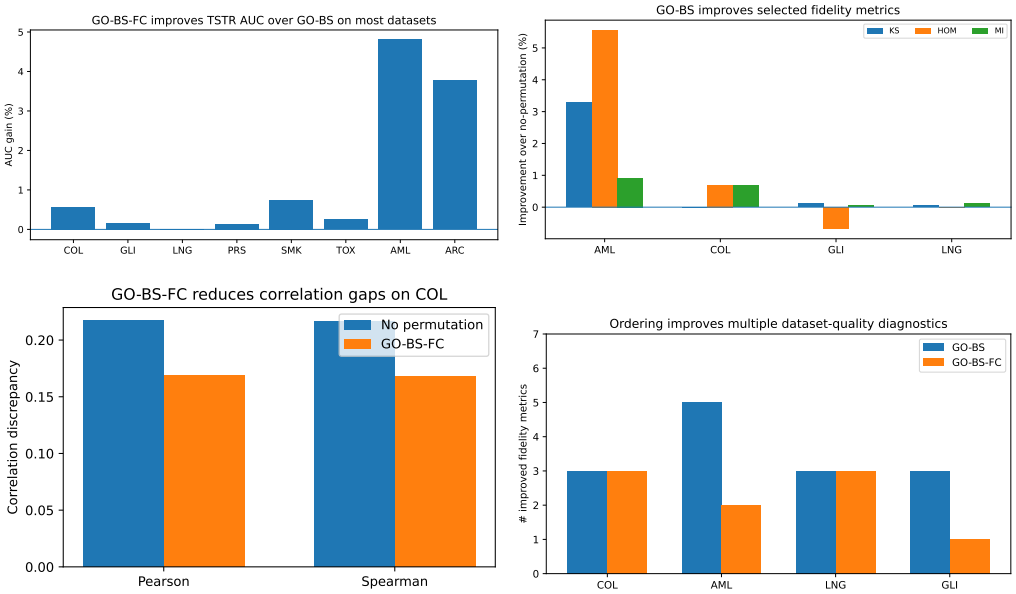


Figure A3.1: **Effect of feature ordering.** GO-BS-FC generally improves TSTR AUC over GO-BS, while GO-BS improves selected fidelity diagnostics over the no-permutation baseline. Ordering also reduces COL correlation discrepancies and improves multiple fidelity metrics across datasets, indicating better structural preservation.

1. GO-BS (sample-clustered; main method). Given training data $X \in \mathbb{R}^{n \times m}$, we first partition samples into C clusters. For each sample cluster c, we construct a feature-affinity graph

$$G_c = (V, E, W^{(c)}), \qquad V = \{1, \ldots, m\} \tag{41}$$

we derive a local feature ordering π_c, and integrate the local orderings into a global order Π^*. The globally ordered feature sequence is then segmented into M contiguous blocks using the GO-BS objective in Eq. 15.

2. GO-BS-FC (feature-clustered ablation). As an alternative, we directly cluster feature vectors by transposing the data matrix and treating each feature as an object in $\mathbb{R}^n$:

$$X^\top \in \mathbb{R}^{m \times n} \tag{42}$$

We partition the m features into C_f clusters in this transposed space, order the feature clusters, and then order features within each cluster using the same graph metric as in the main method. The resulting concatenated order is then segmented into M contiguous blocks.

3. No clustering. As a simpler baseline, we also consider building a single global feature-affinity graph from the full dataset without any sample or feature clustering, followed by one global ordering and contiguous segmentation.

Why Sample Clustering is the Main Design. We choose sample clustering as the main GO-BS design for four reasons.

(1) Local dependence regimes. Sample clustering allows GO-BS to capture regime-specific feature structure. If two features are strongly associated only within a subset of samples, a global feature-clustering approach may weaken that relationship by averaging over the entire dataset. In contrast, a cluster-specific feature graph can preserve such local structure.

(2) Local-to-global ordering. The GO-BS pipeline is designed as a local-to-global ordering mechanism: local feature orderings are derived within sample clusters and then integrated into a single global order. This aggregation step is naturally related to rank aggregation, where one seeks a "globally consistent solution" from potentially inconsistent partial order information (Ailon et al., 2008). In our view, this is better aligned with graph-guided ordering than direct feature clustering, because seriation methods are explicitly concerned with recovering a linear order from similarity information (Hahsler et al., 2008; Fogel et al., 2016), whereas feature clustering primarily yields discrete groups.

(3) Better compatibility with contiguous segmentation. The final step of GO-BS is to partition the global ordered sequence into contiguous blocks for BSTabDiff. Sample-clustered local orderings naturally produce a global rank aggregation problem, which then feeds cleanly into contiguous segmentation. Direct feature clustering tends instead to produce a discrete grouping first, making the subsequent ordering step less central.

(4) Consistency with the intended role of GO-BS. GO-BS is not intended to replace BSTabDiff with a feature-clustering model. Rather, it serves as a permutation-estimation module that recovers a canonical feature order before fitting the original block-subunit generator. The sample-clustered design better supports this interpretation.

Experimental Setup. For this ablation, we keep all BSTabDiff components fixed and vary only the front-end ordering strategy. In particular, the following are held constant across variants:

- the number of latent blocks M,
- the prior type,
- the emission model,
- the latent inference procedure, and
- the training and sampling protocol.

Thus, any performance difference is attributable to the ordering mechanism rather than to changes in the generative model itself. We evaluate the three variants on the Colon dataset using the same downstream protocol as in the main paper. Specifically, we report:

- **synthetic utility** via strict fold-wise 5×5 CV TSTR/TRTR evaluation,
- **distributional fidelity** via per-feature KS/Wasserstein statistics and correlation-structure recovery,
- **label-information preservation** via mutual-information discrepancy, and
- **runtime and memory usage**.

All variants use the same BSTabDiff hyperparameters; only the ordering front-end is changed.

Ablation Results. Table A4.1 reports synthetic utility across four downstream classifiers. The three variants are broadly competitive on Colon, with only modest differences overall. Nevertheless, several trends are visible.

- First, the proposed sample-clustered GO-BS attains the best average TSTR accuracy across the four classifiers (0.8556), slightly ahead of feature clustering (0.8538) and no clustering (0.8533). It also gives the strongest TSTR accuracy for CatBoost and Logistic Regression, while matching the best TabPFN-2.5 result. This indicates that the sample-clustered ordering is a robust default when the goal is downstream synthetic utility across diverse classifier families.

- Second, GO-BS-FC is highly competitive and in some cases yields the best TSTR AUC, especially with TabPFN-2.5. However, its gains are not uniform across classifiers, and its average TSTR accuracy remains marginally below the sample-clustered variant.

- Third, the no-clustering baseline is surprisingly strong in several cases. This suggests that the Colon dataset already contains a fairly coherent global dependence structure, so even a single global affinity graph can recover a useful ordering. As a result, the benefit of sample clustering on this dataset is real but necessarily modest.

- TRTR scores are nearly identical across all three variants, as expected, since the real data classifier pipeline is unchanged and only the synthetic generator front-end differs. Hence, the main signal in this ablation comes from TSTR and fidelity rather than TRTR.

Table A4.2 reports fidelity and resource usage. The picture here is more nuanced. The sample-clustered variant yields the best mean KS discrepancy and remains competitive on Wasserstein distance and mutual-information preservation. The feature-clustered variant achieves the best Pearson/Spearman correlation recovery and is also the fastest variant, reducing training time by more than $2\times$ relative to the sample-clustered pipeline. Meanwhile, no clustering gives the best KS max,

Table A4.1: Synthetic utility of GO-BS ordering variants on Colon under strict fold-wise 5×5 CV. We report TSTR/TRTR mean±std across four downstream classifiers. Sample-clustered GO-BS (+) gives the best average TSTR accuracy, while all variants remain competitive; $*$ denotes feature-clustered GO-BS-FC. Gold highlights the best TSTR values for each classifier and the best aggregate TSTR results.

Variant	Classifier	TSTR Acc. ↑	TSTR AUC ↑	TRTR Acc. ↑	TRTR AUC ↑
GO-BS+	TabPFN-2.5	$0.8715_{\pm 0.0991}$	$0.8988_{\pm 0.1047}$	$0.8464_{\pm 0.0951}$	$0.8825_{\pm 0.1075}$
	TANDEM	$0.8431_{\pm 0.1118}$	$0.8938_{\pm 0.0978}$	$0.8333_{\pm 0.0857}$	$0.8888_{\pm 0.0913}$
	CatBoost	$\mathbf{0.8551}_{\pm 0.1098}$	$0.8770_{\pm 0.1073}$	$0.8246_{\pm 0.1017}$	$\mathbf{0.8920}_{\pm 0.1094}$
	Log. Reg.	$\mathbf{0.8526}_{\pm 0.1074}$	$0.8928_{\pm 0.0999}$	$\mathbf{0.8431}_{\pm 0.1036}$	$0.8703_{\pm 0.1092}$
GO-BS-FC$*$	TabPFN-2.5	$0.8715_{\pm 0.0991}$	$\mathbf{0.9032}_{\pm 0.1008}$	$0.8464_{\pm 0.0951}$	$0.8825_{\pm 0.1075}$
	TANDEM	$\mathbf{0.8459}_{\pm 0.1281}$	$0.8892_{\pm 0.1084}$	$0.8333_{\pm 0.0857}$	$0.8888_{\pm 0.0913}$
	CatBoost	$0.8487_{\pm 0.1089}$	$\mathbf{0.8895}_{\pm 0.0988}$	$0.8246_{\pm 0.1017}$	$\mathbf{0.8920}_{\pm 0.1094}$
	Log. Reg.	$0.8492_{\pm 0.1166}$	$\mathbf{0.8978}_{\pm 0.1006}$	$\mathbf{0.8431}_{\pm 0.1036}$	$0.8703_{\pm 0.1092}$
No clustering	TabPFN-2.5	$0.8715_{\pm 0.0991}$	$0.9017_{\pm 0.1077}$	$0.8464_{\pm 0.0951}$	$0.8825_{\pm 0.1075}$
	TANDEM	$0.8431_{\pm 0.1190}$	$0.8915_{\pm 0.1079}$	$0.8333_{\pm 0.0857}$	$0.8888_{\pm 0.0913}$
	CatBoost	$0.8526_{\pm 0.0994}$	$0.8805_{\pm 0.1079}$	$0.8246_{\pm 0.1017}$	$\mathbf{0.8920}_{\pm 0.1094}$
	Log. Reg.	$0.8459_{\pm 0.1109}$	$0.8938_{\pm 0.0967}$	$\mathbf{0.8431}_{\pm 0.1036}$	$0.8703_{\pm 0.1092}$
Avg.: GO-BS		**0.8556**	0.8906	0.8369	0.8834
Avg.: GO-BS-FC		0.8538	**0.8949**	0.8369	0.8834
Avg.: No clustering		0.8533	0.8919	0.8369	0.8834

Table A4.2: Fidelity and resource comparison for GO-BS ordering variants on Colon. Lower is better for fidelity discrepancies and runtime; higher NN distance suggests less direct copying. Feature clustering is fastest and best for correlation recovery, while sample clustering gives the strongest synthetic utility in Table A4.1. Here, + denotes sample-clustered GO-BS and $*$ denotes feature-clustered GO-BS-FC. Here, KS = Kolmogorov-Smirnov, W1 = Wasserstein-1, MI = mutual information, NN = nearest neighbor, and RSS = CPU resident set size.

Variant	KS mean ↓	KS max ↓	W1 mean ↓	W1 max ↓	Pearson ↓	Spearman ↓
GO-BS+	**0.0931**	0.2960	**0.1375**	0.4842	0.2178	0.2174
GO-BS-FC$*$	0.0978	0.2960	0.1495	0.5216	**0.1807**	**0.1800**
No clustering	0.0943	**0.2848**	0.1397	**0.4484**	0.2137	0.2136

Variant	MI diff ↓	NN dist ↑	Time (s) ↓	CPU RSS (GiB) ↓	GPU (GiB) ↓
GO-BS+	0.2252	**49.9973**	27.30	**1.6231**	0.0357
GO-BS-FC$*$	0.2282	45.7295	**11.97**	1.6279	0.0357
Global ordering	**0.2251**	49.5847	15.03	1.6329	0.0357

W1 max, and marginally best mutual-information discrepancy. Taken together, these results suggest that no single variant dominates every criterion on Colon. Instead, the proposed sample-clustered GO-BS offers the best overall balance for our intended use: it gives the strongest average downstream synthetic utility while remaining competitive on fidelity, at the cost of additional front-end computation. Feature clustering is an efficient alternative when runtime is a stronger concern, whereas no clustering serves as a strong but less expressive baseline.

Discussion. The ablation supports three conclusions.

(1) Sample clustering is a strong default for GO-BS. Although the margins are small on Colon, the proposed sample-clustered variant gives the best average TSTR accuracy across four downstream classifiers. This is the most relevant criterion for our use case, since GO-BS is intended as a front-end permutation estimator for improving the practical utility of BSTabDiff-generated samples.

(2) Feature clustering is efficient but slightly less robust as a default. GO-BS-FC is substantially faster and recovers global correlation structure slightly better, but its downstream gains are less

consistent across classifiers. This is compatible with the interpretation that feature clustering captures a strong global organization, yet is less expressive than sample-conditioned local graphs when local regimes matter.

(3) Colon appears relatively homogeneous. The no-clustering baseline remains surprisingly competitive. This indicates that Colon may already exhibit a coherent global dependence structure, limiting the headroom available to more sophisticated local-to-global ordering strategies. We therefore view this ablation as conservative: on datasets with stronger sample heterogeneity, the relative advantage of sample clustering is expected to be larger.

Overall, the empirical evidence is consistent with our design choice. Even on a dataset where the simpler alternatives are already strong, the proposed sample-clustered GO-BS remains the best-performing overall configuration for downstream synthetic utility, while preserving a competitive fidelity profile.

Practical Note on Implementation. We use GPU acceleration for sample clustering, while feature-graph construction and local refinement are performed on CPU. This design reflects the HDLSS regime: sample clustering benefits from parallel distance computation, whereas dense feature-graph materialization scales quadratically in the number of features and local refinement involves irregular indexing and sequential updates, making a CPU implementation more memory-stable and practical. In our implementation, all three variants used negligible GPU memory for the ordering front-end, and the main computational difference came from the additional clustering and aggregation steps in the sample-clustered pipeline.

Summary. Overall, this ablation isolates an important design choice in GO-BS. The main conclusion is that GO-BS is most naturally formulated as

$$\begin{aligned}\text{sample clustering} &\rightarrow \text{cluster-specific feature graphs} \rightarrow \text{local feature orderings} \\ &\rightarrow \text{global order aggregation} \rightarrow \text{contiguous block segmentation}\end{aligned}$$

whereas direct feature clustering serves as a simpler and faster, but somewhat less expressive, alternative. On Colon, all three variants are competitive, but the sample-clustered design provides the strongest overall downstream synthetic utility and therefore remains the main design used in BSTabDiff.

A5 GO-BS Ordering Ablation against Classical Metaheuristics

We compare the GO-BS family against classical metaheuristic feature ordering modules inside BSTabDiff on Colon. The compared methods are: (i) sample-clustered GO-BS, which derives ordering from sample-cluster-specific feature affinities; (ii) feature-clustered GO-BS-FC; and global explicit orderings from (iii) greedy affinity traversal, (iv) Christofides (Christofides, 2022), (v) Simulated Annealing TSP (Kirkpatrick et al., 1983), (vi) Threshold Accepting TSP (Dueck and Scheuer, 1990), (vii) Genetic Algorithm (Larranaga et al., 1999), and (viii) Ant Colony Optimization (Dorigo and Gambardella, 2002). Christofides, Simulated Annealing TSP, and Threshold Accepting TSP are implemented using NetworkX (Hagberg et al., 2008). For all explicit-order baselines, we construct the same cosine-based top-k feature-affinity graph from the training split, convert affinities to distances, compute a permutation, and fit BSTabDiff with this fixed external ordering while disabling internal GO-BS. For GO-BS and GO-BS-FC, we report diagnostics from explicit materializations of their ordering procedures.

Evaluation follows the strict 5×5 repeated stratified cross-validation protocol of Jiang et al. (2024), using TabPFN-2.5 (Grinsztajn et al., 2025) as the downstream evaluator. On each fold, BSTabDiff is trained on the real training split, generates $n_{\text{syn}} = 200$ samples, and TabPFN is trained on synthetic data and tested on the real held-out fold (TSTR). We also report the real data TabPFN baseline on the same folds (TRTR), which is identical across rows because it does not depend on generator ordering. Beyond utility, we measure ordering time, total training time, GPU peak memory, CPU RSS, TSP-path cost, Minimum Linear Arrangement (MinLA) cost, and adjacent-edge locality reward; lower TSP/MinLA costs and higher locality reward are preferred.

Table A5.1 shows that downstream differences across orderings are modest, indicating that BSTabDiff is robust when the permutation remains structure-aware. Still, the GO-BS variants are among the strongest: GO-BS-FC gives the best mean TSTR accuracy (0.8715 ± 0.0991), greedy global ordering gives the best TSTR AUC (0.9087 ± 0.0925), and sample-clustered GO-BS remains competitive

Table A5.1: GO-BS ablation against classical metaheuristic orderings on Colon. Results are mean$_{\pm \text{std}}$ over strict 5×5 repeated stratified CV using TabPFN-2.5 as the downstream evaluator. TSTR trains on BSTabDiff synthetic data and tests on real held-out folds; TRTR is the real data reference on the same splits. Gold, silver, and bronze indicate the top three distinct TSTR values (ties share colors); best values are bolded. GO-BS/GO-BS-FC denote sample-/feature-clustered variants; Greedy = global ordering; Christ. = Christofides; SA/TA-TSP = Simulated Annealing/Threshold Accepting TSP; GA = Genetic Algorithm; ACO = Ant Colony Optimization.

Method	TSTR Acc. ↑	TSTR AUC ↑	TRTR Acc. ↑	TRTR AUC ↑
GO-BS	$0.8682_{\pm 0.0989}$	$0.9030_{\pm 0.1000}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
GO-BS-FC	$\mathbf{0.8715}_{\pm 0.0991}$	$0.9073_{\pm 0.1009}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
Greedy	$0.8651_{\pm 0.1053}$	$\mathbf{0.9087}_{\pm 0.0925}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
Christ.	$0.8682_{\pm 0.0989}$	$0.8972_{\pm 0.1124}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
SA-TSP	$0.8618_{\pm 0.1076}$	$0.9035_{\pm 0.0983}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
TA-TSP	$0.8618_{\pm 0.1076}$	$0.9035_{\pm 0.0983}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
GA	$0.8682_{\pm 0.0989}$	$0.8982_{\pm 0.1040}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
ACO	$0.8615_{\pm 0.0982}$	$0.9055_{\pm 0.1033}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$

Table A5.2: Ordering and resource diagnostics for the GO-BS ablation on Colon. Lower is better for ordering time, total training time, GPU peak, CPU RSS, TSP-path cost, and MinLA cost; higher is better for edge-locality reward (ELR). GO-BS and GO-BS-FC diagnostics use explicit materializations of their ordering procedures. Gold, silver, and bronze indicate the three best distinct values in each metric; ties share colors. GO-BS/GO-BS-FC denote sample/feature clustered variants; Greedy = global ordering; Christ. = Christofides; SA/TA-TSP = Simulated Annealing/Threshold Accepting TSP; GA = Genetic Algorithm; ACO = Ant Colony Optimization; Ord. = ordering; RSS = CPU resident set size.

Method	Ord. Time (s) ↓	Train Time (s) ↓	GPU Peak (GiB) ↓	CPU RSS (GiB) ↓	TSP Cost ↓	MinLA Cost ↓	ELR ↑
GO-BS	23.57	28.33	**0.0298**	1.5808	1973.63	4.52e6	25.37
GO-BS-FC	**1.08**	**15.60**	0.0357	1.5026	784.66	**1.62e6**	1214.34
Greedy	4.59	21.24	0.0357	1.5566	737.02	4.47e6	1261.98
Christ.	1951.85	1967.20	0.0357	2.7491	**658.99**	3.79e6	**1340.00**
SA-TSP	804.64	813.60	0.0357	1.4764	693.84	4.08e6	1305.16
TA-TSP	818.23	828.37	0.0357	**1.4749**	693.84	4.08e6	1305.16
GA	820.81	836.15	0.0357	1.4759	789.57	4.42e6	1209.43
ACO	887.01	899.99	0.0357	1.4768	660.24	4.20e6	1338.76

(0.8682 ± 0.0989 accuracy, 0.9030 ± 0.1000 AUC). Heavier TSP-style metaheuristics and ACO do not yield clear utility gains despite more expensive combinatorial search, suggesting that optimizing generic routing surrogates alone is insufficient for synthetic data utility.

Table A5.2 reinforces this conclusion. Christofides attains the best TSP-path cost and one of the best locality rewards, but is far more expensive in ordering and total training time and does not improve downstream utility over the GO-BS variants. GO-BS-FC is especially favorable, combining the lowest MinLA cost (1.62×10^6) with the best TSTR accuracy while remaining lightweight (1.08s ordering, 15.60s total training). Conversely, sample-clustered GO-BS has weaker generic surrogate scores but remains competitive in TSTR, showing that TSP cost, MinLA, and adjacent-edge reward do not fully explain synthetic data usefulness. Overall, the GO-BS family appears effective not merely because it produces a permutation, but because it imposes data-aware structural priors that better preserve task-relevant feature organization for HDLSS generation and downstream inference.

The ablation therefore raises a model-selection question: if GO-BS (sample-clustered) is competitive but not uniformly best on every Colon-specific summary statistic, why do we retain it as the canonical formulation in the main experiments? Our answer is that the canonical choice should not be made by a post hoc weighted average of heterogeneous surrogates. Instead, we adopt a two-stage principle that first filters methods by downstream utility and then selects among near-equivalent methods using structural alignment with BSTabDiff's intended inductive bias.

Definition A5.1 (Structure-aligned selection rule)**.** For a candidate ordering method q, we define downstream utility in Eq. 43:

$$U(q) = \frac{1}{2}\big(\mathrm{TSTRAcc}(q) + \mathrm{TSTRAUC}(q)\big) \tag{43}$$

Define the maximum utility and near-optimal candidate set by Eq. 44:

$$U^\star = \max_{q' \in \mathcal{Q}} U(q'), \quad \mathcal{A}_\varepsilon \quad = \{q \in \mathcal{Q} : U^\star - U(q) \le \varepsilon\} \tag{44}$$

We select the canonical method by Eq. 45:

$$q^\dagger = \arg\min_{q \in \mathcal{A}_\varepsilon} \mathcal{R}_{\mathrm{align}}(q) \tag{45}$$

where $\mathcal{R}_{\mathrm{align}}(q)$ is a structure-aligned risk measuring compatibility between the induced ordering and BSTabDiff's regime-aware contiguous block assumption.

Definition A5.2 (Regime-aware alignment risk)**.** Let the training samples be partitioned into K_s sample clusters with weights $\alpha_c = n_c/n$. For a candidate ordering-segmentation pair $(\pi, \mathbf{b})$, we define Eq. 46.

$$\mathcal{R}_{\mathrm{align}}(\pi, \mathbf{b}) = \sum_{c=1}^{K_s} \alpha_c \, \mathcal{J}^{(c)}_{\text{GO-BS}}(\pi, \mathbf{b}) + \eta \, \mathrm{Var}_{c=1,\dots,K_s}\Big(\mathcal{J}^{(c)}_{\text{GO-BS}}(\pi, \mathbf{b})\Big) \tag{46}$$

where $\mathcal{J}^{(c)}_{\text{GO-BS}}$ is the GO-BS block-consistency objective computed on the cluster-specific feature affinity graph of sample cluster c. Lower values are better.

Remark A5.3. Classical metaheuristics such as Christofides, simulated annealing, threshold accepting, genetic algorithm, and ant colony optimization optimize global route-style surrogates on a single affinity graph, whereas GO-BS (sample-clustered) optimizes a regime-aware block-consistency criterion directly aligned with BSTabDiff's intended latent block structure. Hence, when downstream utility differences are practically small, GO-BS (sample-clustered) is the methodologically preferred canonical choice.

Although sample-clustered GO-BS is not uniformly best on every Colon-specific statistic, we do not select the canonical ordering rule by a post hoc aggregate of heterogeneous surrogates. Instead, we use a two-stage principle: first, retain methods with practically comparable downstream utility; second, among these, prefer the method most aligned with BSTabDiff's regime-aware block-factorized inductive bias. Under this criterion, generic TSP-style proxies, runtime, MinLA, and edge-locality reward are treated as diagnostics rather than the sole selection objective.

Sample-clustered GO-BS is the only variant that directly optimizes a regime-aware block-consistency risk aggregated over sample-local feature graphs. This matches the intended role of BSTabDiff: recovering a canonical feature order in which locally coherent, sample-conditioned dependencies become contiguous and segmentable into latent blocks. In contrast, the classical metaheuristics optimize route or arrangement surrogates on a single global graph, while GO-BS-FC replaces the sample-conditioned local-to-global mechanism with feature clustering. Since these alternatives do not deliver a decisive downstream gain on Colon, we retain sample-clustered GO-BS as the canonical formulation: it is the structurally aligned, theory-consistent ordering rule for BSTabDiff while remaining empirically competitive.

A6 Ordering-Solver Ablation on a Fixed Global Affinity Graph

Table A6.1 isolates the effect of the ordering solver by fixing a single global feature-affinity graph and varying only the permutation procedure within BSTabDiff. Under this controlled setting, downstream utility differences are again modest, indicating that BSTabDiff is reasonably robust to the exact solver once a structure-aware global graph is available. Among the fixed-graph solvers, greedy and ACO achieve the strongest TSTR AUC values (0.9110 and 0.9075, respectively), while Christofides and GA attain the best TSTR accuracy (0.8682). However, Table A6.2 shows that these generic route-optimization methods often incur substantial computational overhead: Christofides attains

Table A6.1: **Same global graph ordering ablation in BSTabDiff on Colon.** Results are mean$_{\pm\text{std}}$ over strict 5×5 repeated stratified cross-validation using TabPFN-2.5 as the downstream evaluator. For the main ablation rows, a single global feature-affinity graph is fixed and only the ordering solver is changed. TSTR trains on synthetic data generated by BSTabDiff and tests on real held-out folds; TRTR is the real data TabPFN reference on the same splits. GO-BS and GO-BS-FC are reported only as reference methods from the earlier structural ablation and are therefore not part of the identical-protocol fixed-graph comparison. Gold, silver, and bronze indicate the top three distinct values among the fixed-graph ordering solvers for the TSTR metrics. TRTR values are identical across fixed-graph solvers and are therefore not ranked. *Abbreviations:* Greedy = Global graph-guided greedy; Christ. = Christofides; SA-TSP = Simulated Annealing TSP; TA-TSP = Threshold Accepting TSP; GA = Genetic Algorithm; ACO = Ant Colony Optimization; TSTR = Train on Synthetic, Test with Real; TRTR = Train on Real, Test with Real.

Method	TSTR Acc. ↑	TSTR AUC ↑	TRTR Acc. ↑	TRTR AUC ↑
GO-BS†	$0.8682_{\pm 0.0989}$	$0.9030_{\pm 0.1000}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
GO-BS-FC†	$0.8715_{\pm 0.0991}$	$0.9073_{\pm 0.1009}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
Greedy	$0.8651_{\pm 0.1053}$	$0.9110_{\pm 0.0933}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
Christ.	$0.8682_{\pm 0.0989}$	$0.8972_{\pm 0.1124}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
SA-TSP	$0.8585_{\pm 0.1045}$	$0.9035_{\pm 0.0983}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
TA-TSP	$0.8585_{\pm 0.1045}$	$0.9035_{\pm 0.0983}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
GA	$0.8682_{\pm 0.0989}$	$0.8993_{\pm 0.1007}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$
ACO	$0.8649_{\pm 0.0958}$	$0.9075_{\pm 0.1034}$	$0.8433_{\pm 0.0987}$	$0.8838_{\pm 0.1072}$

†Reference only: GO-BS and GO-BS-FC come from the earlier structural ablation and do not share the same fixed-global-graph protocol as the other rows.

Table A6.2: **Ordering and resource diagnostics for the same global graph ordering ablation on Colon.** For the main ablation rows, all methods use the same fixed global feature affinity graph and differ only in the ordering solver. Lower is better for ordering time, total train time, GPU peak, CPU RSS, TSP path cost, and MinLA cost; higher is better for edge locality reward (ELR). GO-BS and GO-BS-FC are included as reference points from the earlier structural ablation and are not part of the identical-protocol fixed-graph comparison. Gold, silver, and bronze indicate the top three distinct values among fixed-graph ordering solvers (ties share colors); best values are bolded. *Abbreviations:* Greedy = Global graph-guided greedy; Christ. = Christofides; SA-TSP = Simulated Annealing TSP; TA-TSP = Threshold Accepting TSP; GA = Genetic Algorithm; ACO = Ant Colony Optimization; Ord. = Ordering; ELR = Edge-Locality Reward; RSS = Resident Set Size.

Method	Ord. Time (s) ↓	Train Time (s) ↓	GPU Peak (GiB) ↓	CPU RSS (GiB) ↓	TSP Cost ↓	MinLA Cost ↓	ELR ↑
GO-BS†	23.57	28.33	0.0298	1.5808	1973.63	4.52e6	25.37
GO-BS-FC†	1.08	15.60	0.0357	1.5026	784.66	1.62e6	1214.34
Greedy	**4.38**	**14.45**	**0.0297**	**1.1861**	737.02	4.47e6	1261.98
Christ.	2151.66	2161.77	0.0357	2.4844	**658.99**	**3.79e6**	**1340.00**
SA-TSP	848.24	857.62	0.0357	1.2278	693.84	4.08e6	1305.16
TA-TSP	792.34	802.01	0.0357	1.2279	693.84	4.08e6	1305.16
GA	800.18	809.49	0.0357	1.2293	789.57	4.42e6	1209.43
ACO	892.25	901.96	0.0357	1.2288	660.24	4.20e6	1338.76

†Reference only: GO-BS and GO-BS-FC come from the earlier structural ablation and do not share the same fixed-global-graph protocol as the other rows.

the best TSP-path cost and ELR, but is orders of magnitude slower than greedy, while SA-TSP, TA-TSP, GA, and ACO remain far more expensive without yielding decisive downstream gains. In contrast, the simple global greedy solver is the most practical fixed-graph method, combining the best TSTR AUC with by far the lowest runtime and memory footprint among the same-protocol solvers. When compared against the reference rows, GO-BS and especially GO-BS-FC remain competitive or stronger overall despite not belonging to the identical fixed-graph protocol, reinforcing our main conclusion that BSTabDiff benefits more from structurally informed ordering mechanisms than from heavier optimization of generic TSP-style surrogates alone.

A7 Ablation of GO-BS in Extreme Dimensionality

To study GO-BS under extreme dimensionality, we performed an ordering and training diagnostic ablation on the Cell Cycle dataset ($n = 1067$, $m = 42{,}728$, $C = 3$)(Mahdessian et al., 2021; NCBI,

Table A7.1: **GO-BS diagnostics under extreme dimensionality on Cell Cycle.** The dataset has $n = 1067$, $m = 42{,}728$, and $C = 3$. Lower is better for ordering time, total training time, GPU peak, CPU RSS, TSP-path cost, and MinLA; higher is better for edge-locality reward (ELR). GO-BS/GO-BS-FC denote sample-/feature-clustered variants; Greedy = global greedy ordering; Ord. = ordering; RSS = resident set size.

Method	Ord. Time (s) ↓	Train Time (s) ↓	GPU Peak (GiB) ↓	CPU RSS (GiB) ↓	TSP Cost ↓	MinLA Cost ↓	ELR ↑
GO-BS	43629.60	21398.20	0.8600	**29.4335**	42655.0	1.64078e9	71.9512
GO-BS-FC	**3798.98**	69.7264	**0.4431**	56.7750	35117.6	**1.62298e9**	7609.40
Greedy	6727.97	**64.5725**	**0.4431**	83.9261	**32878.3**	1.87887e9	**9848.66**

2021), comparing the main sample-clustered GO-BS, the feature-clustered GO-BS-FC variant, and a simple global greedy ordering baseline. The results in Table A7.1 show a clear computational separation between the methods: the main sample-clustered GO-BS becomes extremely expensive in this regime, requiring substantially larger ordering and training times, whereas GO-BS-FC remains much more practical while also achieving the best MinLA cost among the three methods. Although the global greedy baseline attains the lowest TSP cost and the highest edge-locality reward, it does so with a worse MinLA score and much larger CPU memory usage. Overall, this ablation suggests that in extreme dimensional settings, GO-BS-FC provides the strongest trade-off between structural quality and computational feasibility, while the full sample-clustered GO-BS may become prohibitively expensive.

A8 BSTabDiff in Extreme Dimensionality

To further evaluate BSTabDiff under extreme dimensionality, we consider the Cell Cycle RNA-seq dataset (Mahdessian et al., 2021; NCBI, 2021), which contains $1{,}067$ samples and $42{,}728$ features across three classes. This setting is substantially larger in feature dimension than the classical HDLSS benchmarks and directly tests whether the block-subunit formulation remains practical when m is extremely large. Since TabPFN-2.5 (Grinsztajn et al., 2025) is not applicable in this regime due to its feature-limit constraint, we report TSTR using Logistic Regression as a fixed downstream classifier. Table A8.1 compares BSTabDiff without GO-BS, with sample-clustered GO-BS, and with feature-clustered GO-BS-FC.

The results show that BSTabDiff remains effective even at $m = 42{,}728$ features, achieving TSTR AUCs above 91% for all variants. GO-BS-FC gives the strongest utility, improving TSTR accuracy from 77.65% to 80.20% and TSTR AUC from 91.40% to 92.83%, while also producing the lowest C2ST accuracy/AUC, suggesting slightly harder real-versus-synthetic discrimination. In contrast, sample-clustered GO-BS does not improve TSTR on this dataset and is substantially more expensive, increasing peak GPU memory to 27.554 GiB because the sample-clustered feature-affinity construction operates over very large feature graphs. Thus, for extreme dimensional RNA-seq data, GO-BS-FC provides the best trade-off: it preserves the block-structured inductive bias of GO-BS while avoiding the quadratic GPU-memory burden of constructing dense sample-clustered feature-affinity graphs over all $42{,}728$ features.

Table A8.2 shows that ordering does not uniformly improve all low-level distributional fidelity metrics in this extreme dimensional setting. The marginal KS statistics are nearly identical across all variants, while no-GO-BS has the lowest Wasserstein-1 (W1) mean. Sample-clustered GO-BS substantially improves variance matching, reducing the mean variance discrepancy from 1.5798×10^3 to 1.0716×10^3, but it slightly worsens Spearman and MI discrepancies and requires much larger GPU memory. GO-BS-FC gives the best downstream synthetic utility and the lowest C2ST scores, while also slightly improving feature-label MI preservation and nearest neighbor privacy distance. Thus, the main benefit of GO-BS-FC is not uniformly better marginal fidelity, but a better utility-realism-compute trade-off under extreme dimensionality.

Overall, BSTabDiff with GO-BS-FC provides the best utility realism compute trade-off in the extreme dimensional setting. On Cell Cycle ($n = 1067, m = 42{,}728$), it improves TSTR AUC from 91.40% to 92.83% and reduces C2ST AUC from 58.28% to 54.57%, while requiring nearly the same peak GPU memory as no-GO-BS. In contrast, sample-clustered GO-BS becomes memory-intensive at this scale, reaching 27.554 GiB peak GPU memory without improving TSTR. Although GO-BS-FC does not uniformly dominate all marginal fidelity metrics, these results suggest that feature-clustered blocking better preserves task-relevant structure and is the more scalable ordering strategy for very high-dimensional RNA-seq tables.

Table A8.1: BSTabDiff on extreme dimensional Cell Cycle data ($n = 1{,}067$, $m = 42{,}728$, $C = 3$). All variants use Optuna-selected hyperparameters without feature permutation and are evaluated with Logistic Regression under strict 5×5 TSTR/TRTR/C2ST. Accuracy and AUC are percentages; peak GPU is max CUDA memory and CPU memory is end-of-run RSS on Vulcan.

Variant	TSTR Acc. (%)	TSTR AUC (%)	TRTR Acc. (%)	TRTR AUC (%)	C2ST Acc. (%)	C2ST AUC (%)	Time (s)	GPU/CPU (GiB)
No GO-BS	77.7±4.4	91.4±2.2	86.7±2.7	95.4±1.1	55.8±10.9	58.3±13.8	17.55	0.635/3.723
GO-BS	77.3±4.4	91.4±2.3	86.7±2.7	95.4±1.1	56.5±9.6	58.9±13.7	101.15	27.554/1.934
GO-BS-FC	**80.2**±5.2	**92.8**±2.0	86.7±2.7	95.4±1.1	**53.7**±12.8	**54.6**±15.5	22.11	0.635/3.657

Table A8.2: Fidelity diagnostics for BSTabDiff variants on Cell Cycle. Correlation and MI are computed on a fixed 2,000-feature subsample. Lower is better for all discrepancies except NN distance, where higher values suggest less direct memorization.

Variant	KS Mean	W1 Mean	Spearman Diff	ΔMI Mean	Mean Diff	Var Diff	NN Dist. Mean
No GO-BS	0.3875	0.8933	0.3700	0.0323	4.6870×10^{-1}	1.5798×10^{3}	6.6940×10^{3}
GO-BS	0.3876	0.8957	0.3727	0.0325	4.8673×10^{-1}	$\mathbf{1.0716\times10^{3}}$	6.6605×10^{3}
GO-BS-FC	0.3880	1.4628	0.3710	**0.0320**	5.3319×10^{-1}	1.5275×10^{3}	$\mathbf{6.7050\times10^{3}}$

A9 Energy Use, Carbon Emissions, and Green AI Considerations

Training a single AI model can emit as much carbon as five cars in their lifetimes (Hao, 2019). In modern GPU clusters, hardware-level resource utilization can be monitored using tools such as `nvidia-smi`, which reports GPU memory usage, process-level GPU allocation, instantaneous power draw in watts, and device-level information such as temperature and fan activity. We additionally use `psutil` (Rodola, 2026) to record CPU-side memory usage and system resource statistics during model execution. Since carbon emissions are commonly estimated from energy consumption and the carbon intensity of the electricity grid, a model that completes the same task with lower power draw or shorter runtime generally yields lower emissions under a fixed grid mix (Lannelongue et al., 2021; Lacoste et al., 2019). This relationship motivates recent work on Green AI (Schwartz et al., 2020), Sustainable AI (Tabbakh et al., 2024), and energy efficient machine learning (e.g. IMLP (Wang et al., 2025b)), where models are evaluated not only by predictive performance but also by computational cost, energy consumption, and environmental impact (Strubell et al., 2019; Bolón-Canedo et al., 2024). At the AI-power grid interface, Nwakanma and Srivastava (2026) further emphasize energy-aware AI design and transparent reporting of training and inference energy consumption and associated carbon emissions. In this context, energy-aware model design encourages efficient architectures and training strategies that reduce unnecessary computation while maintaining competitive performance, supporting the broader goal of developing machine learning systems with lower operational energy demand and reduced carbon footprint (Schwartz et al., 2020; Lannelongue et al., 2021; Khan et al., 2025). We further introduce computational sustainability as an additional evaluation criterion for tabular generative models. Beyond assessing synthetic data quality through downstream utility and distributional fidelity, we track resource usage during training, estimate total energy consumption, and derive the corresponding carbon-emission estimate. This follows the Green AI perspective, which argues that model quality should be interpreted jointly with computational cost rather than predictive performance alone (Schwartz et al., 2020; Strubell et al., 2019). Under this criterion, a tabular generator is preferable when it achieves comparable synthetic data utility with lower energy consumption, since carbon emissions are commonly estimated from energy use and grid carbon intensity; therefore, under a fixed electricity mix, lower energy demand generally implies lower emissions (Lannelongue et al., 2021; Anthony et al., 2020; Lacoste et al., 2019).

Practical measurement setup. During training, we monitor GPU telemetry with `nvidia-smi`, including memory usage, process-level allocation, temperature, fan speed, and instantaneous power draw. We use `psutil` (Rodola, 2026) to record CPU-side memory and system resource statistics. We also use Python-based tools such as CarbonTracker (Anthony et al., 2020) and CodeCarbon (Courty et al., 2026) to estimate energy consumption and carbon emissions during model training (Lacoste et al., 2019).

Power and energy formulation. Let $P_{\mathrm{GPU}}(t)$ denote the instantaneous GPU power draw (in watts) at time t, and let $P_{\mathrm{CPU}}(t)$ denote the corresponding CPU power draw if available. If measurements are collected at discrete times $t_1, \ldots, t_T$, then the total training energy can be approximated by Eq. 47.

$$E_{\mathrm{train}} \approx \sum_{i=1}^{T-1} \big(P_{\mathrm{GPU}}(t_i) + P_{\mathrm{CPU}}(t_i)\big)\, \Delta t_i \tag{47}$$

where $\Delta t_i = t_{i+1} - t_i$ is measured in hours, so that E_{train} is expressed in watt-hours (Wh). Equivalently, in integral form, we get Eq. 48.

$$E_{\mathrm{train}} = \int_0^{T_{\mathrm{train}}} \big(P_{\mathrm{GPU}}(t) + P_{\mathrm{CPU}}(t)\big)\, dt \tag{48}$$

If only average GPU power is available, one may use the simpler approximation using Eq. 49.

$$E_{\mathrm{GPU}} \approx \bar{P}_{\mathrm{GPU}} \cdot T_{\mathrm{train}} \tag{49}$$

where $\bar{P}_{\mathrm{GPU}}$ is the average GPU power (W) and T_{train} is training time in hours.

Including data-center overhead. A more complete estimate may account for infrastructure overhead through Power Usage Effectiveness (PUE), yielding Eq. 50.

$$E_{\mathrm{total}} = \mathrm{PUE} \times E_{\mathrm{train}} \tag{50}$$

where $\mathrm{PUE} \geq 1$ captures additional cooling and facility-level energy costs (Lannelongue et al., 2021; Hölbling et al., 2025).

Carbon-emission estimate. Let I_{grid} denote the carbon intensity of the electricity supply in units of kg $\mathrm{CO_2e}$/kWh. Then the estimated operational carbon footprint of training is calculated using Eq. 51.

$$\mathrm{CO_2e} = E_{\mathrm{total}}^{(\mathrm{kWh})} \times I_{\mathrm{grid}} \tag{51}$$

where $E_{\mathrm{total}}^{(\mathrm{kWh})}$ is the total energy converted to kWh. Thus, if two models are trained on the same hardware under the same electricity grid mix, the model with lower total energy consumption will yield lower estimated $\mathrm{CO_2e}$ emissions (Lannelongue et al., 2021; Anthony et al., 2020; Lacoste et al., 2019).

Interpretation in terms of Green AI. A model should not be considered "green" solely because it exhibits low instantaneous memory usage or low peak power draw. Rather, from the Green AI perspective, a method is more sustainable when it achieves a favorable trade-off between predictive utility and computational cost while reducing total energy consumption, estimated carbon emissions, or both (Schwartz et al., 2020; Tabbakh et al., 2024; Bolón-Canedo et al., 2024). In practice, useful indicators include: (i) lower training energy in Wh or kWh, (ii) lower estimated $\mathrm{CO_2e}$, (iii) comparable or improved predictive performance per unit energy, (iv) efficient hardware utilization, and (v) transparent reporting of runtime, hardware configuration, energy-measurement methodology, and emissions assumptions. Since there is no single universal numeric threshold for Green AI, the concept is typically operationalized through comparative efficiency, emissions-aware reporting, and environmentally informed model design (Schwartz et al., 2020; Lannelongue et al., 2021; Tabbakh et al., 2024; Bolón-Canedo et al., 2024; Khan et al., 2025). Recent work has further extended sustainability analysis to specific AI application domains, such as recommender systems, and to broader discussions of the environmental costs of rapid AI innovation (Wegmeth et al., 2025; Agarwal and Chakraborti, 2025).

Context. Prior work has shown that the carbon cost of large-scale AI training can be substantial. For example, Strubell et al. (2019) estimated that certain large NLP model-development pipelines can emit carbon at a scale comparable to the lifetime emissions of multiple automobiles. While this does not imply that every machine learning experiment is environmentally costly to the same extent, it motivates reporting energy consumption and estimated emissions as part of model evaluation.

Energy-Carbon Pareto Selection. To jointly evaluate synthetic utility and resource efficiency, we define an energy-aware Pareto criterion over the set of tabular generative models. For each generator $g \in \mathcal{G}$, let A_g denote the mean TSTR accuracy across datasets, E_g denote the total measured energy consumption in kWh, and C_g denote the total CO_2e emission in kg. Since higher TSTR accuracy is preferred while lower energy and carbon cost are preferred, generator g_i is said to dominate generator g_j if it satisfies Eq. 52

$$A_{g_i} \geq A_{g_j}, \qquad E_{g_i} \leq E_{g_j} \tag{52}$$

with at least one strict inequality. Equivalently, when carbon is also included as an objective, dominance requires Eq. 53.

$$A_{g_i} \geq A_{g_j}, \qquad E_{g_i} \leq E_{g_j}, \qquad C_{g_i} \leq C_{g_j} \tag{53}$$

again with at least one strict improvement. The energy-carbon Pareto frontier is therefore the set of non-dominated generators by Eq. 54:

$$\mathcal{P}_{\mathrm{EC}} = \{g \in \mathcal{G} : \nexists g' \in \mathcal{G} \text{ s.t. } A_{g'} \geq A_g,\ E_{g'} \leq E_g,\ C_{g'} \leq C_g, \text{ and at least one inequality is strict}\} \tag{54}$$

In our evaluation, BSTabDiff is the only non-dominated method: it achieves the highest mean TSTR accuracy while simultaneously requiring the lowest total energy consumption and producing the lowest total CO_2e emissions. Thus, BSTabDiff is selected as the preferred Pareto-optimal operating point.

Interpretation of Energy and Carbon Results. The dimensionality-energy (Fig. 3a) and dimensionality-CO_2e (Fig. 3b) plots reveal clear differences in the resource profiles of tabular generative models. CTGAN (Xu et al., 2019) and TVAE (Xu et al., 2019) exhibit the largest energy and carbon footprints, especially as feature dimensionality increases. BinaryDiff (Kinakh and Voloshynovskiy, 2024) also shows substantially higher resource consumption than the more lightweight baselines. Classical and lower-cost methods such as SMOTE (Chawla et al., 2002), TabDDPM (Kotelnikov et al., 2023), TabDiff (Shi et al., 2025), and CtrTab (Li et al., 2026) are generally more efficient, but they do not achieve the same synthetic utility as BSTabDiff. BSTabDiff occupies the most favorable region of the trade-off space: it achieves the highest mean TSTR accuracy while maintaining the lowest measured energy and CO_2e emissions across the benchmark suite. This indicates that the proposed block-subunit design is not only beneficial for HDLSS tabular synthesis quality, but also computationally efficient and environmentally safer.

Pareto Trade-off Analysis for Energy and CO_2e. The Pareto plot (Fig. 3c) compares each method using total energy consumption on the horizontal axis and mean TSTR accuracy on the vertical axis, with bubble size representing total CO_2e emissions. A desirable method should move toward the upper-left region of the plot: higher synthetic utility and lower energy cost. A method is Pareto-efficient if no other method achieves both higher accuracy and lower energy. Under this criterion, BSTabDiff is the sole Pareto-efficient method in our results. Although methods such as SMOTE (Chawla et al., 2002) and TabDiff (Shi et al., 2025) obtain competitive TSTR accuracy, they consume more energy than BSTabDiff while also achieving lower mean TSTR accuracy. Similarly, TVAE (Xu et al., 2019) and CTGAN (Xu et al., 2019) incur substantially larger energy and carbon costs without improving utility. Therefore, the Pareto analysis provides a compact summary of the utility-sustainability trade-off and identifies BSTabDiff as the best overall operating point. Table A9.1 shows that BSTabDiff achieves the highest mean TSTR accuracy while using the lowest total energy and CO_2e, making it the only Pareto-efficient method among the evaluated baselines. We selected the strongest representative baselines for the Pareto-frontier analysis, but observed that GAN-based methods were computationally expensive and that ForestDiffusion (Jolicoeur-Martineau et al., 2024) was by far the heaviest overall; even under a light configuration, its dataset-wise energy/CO_2e/time costs were COL: 0.039 kWh, 0.035 kg, 646.81 s; LNG: 0.205 kWh, 0.184 kg, 3235.18 s; GLI: 4.625 kWh, 4.145 kg, 71755.59 s; SMK: 4.856 kWh, 4.351 kg, 74935.06 s; AML: 0.457 kWh, 0.410 kg, 7405.28 s; PRS: 0.414 kWh, 0.371 kg, 6741.68 s; ARC: 1.172 kWh, 1.051 kg, 18638.22 s; and TOX: 0.752 kWh, 0.674 kg, 11709.77 s, so we excluded ForestDiffusion (Jolicoeur-Martineau et al., 2024), CTAB-GAN (Zhao et al., 2021), and CTAB-GAN+ (Zhao et al., 2024) from the Pareto-frontier analysis to avoid visually compressing the remaining baselines.

Energy and carbon footprint analysis for BSTabDiff. Fig. A9.1 summarizes the energy and carbon footprint of BSTabDiff across four configurations. The Optuna-tuned no-ordering setting is

Table A9.1: Energy-carbon Pareto summary across HDLSS datasets. Mean TSTR accuracy is reported in percentage. Energy is measured in kWh and carbon emission in kg CO_2e. Measurements are obtained using CodeCarbon (Courty et al., 2026). Light green marks the best predictive performance, light blue marks the lowest energy and carbon footprint, and bold Pareto indicators identify non-dominated models.

Model	Mean TSTR Acc. (%)	Total Energy (kWh)	Total CO_2e (kg)	#Datasets	Pareto-efficient
BSTabDiff	**86.33**	**0.003816**	**0.003420**	8	**Yes**
CtrTab	45.77	0.029485	0.026424	8	No
BinaryDiff	71.69	0.266292	0.238648	8	No
TabDiff	81.91	0.020414	0.018295	8	No
TabDDPM	65.67	0.017491	0.015675	8	No
SMOTE	81.80	0.010911	0.009778	8	No
TVAE	80.38	2.274939	2.038773	8	No
CTGAN	57.64	3.709245	3.324182	8	No

highly efficient, with most datasets requiring only 10^{-4}-10^{-3} kWh and sub-10^{-3} kg CO_2e, even for high-dimensional HDLSS data. The fixed/default no-GO-BS setting increases energy moderately, mainly due to the longer 10k-epoch training schedule, but remains in a low absolute carbon range. Adding GO-BS introduces extra structure-learning cost, especially on high-dimensional datasets such as GLI and SMK. GO-BS-FC is usually competitive but shows a clear PRS-specific outlier, suggesting sensitivity of feature-clustered ordering to dataset geometry and clustering cost. Overall, the plots show a clear trade-off: ordering-based structure can improve utility and dataset quality (Fig. A3.1) but adds computational overhead, with GO-BS providing more controlled cost and GO-BS-FC offering stronger structural bias at the risk of occasional energy outliers.

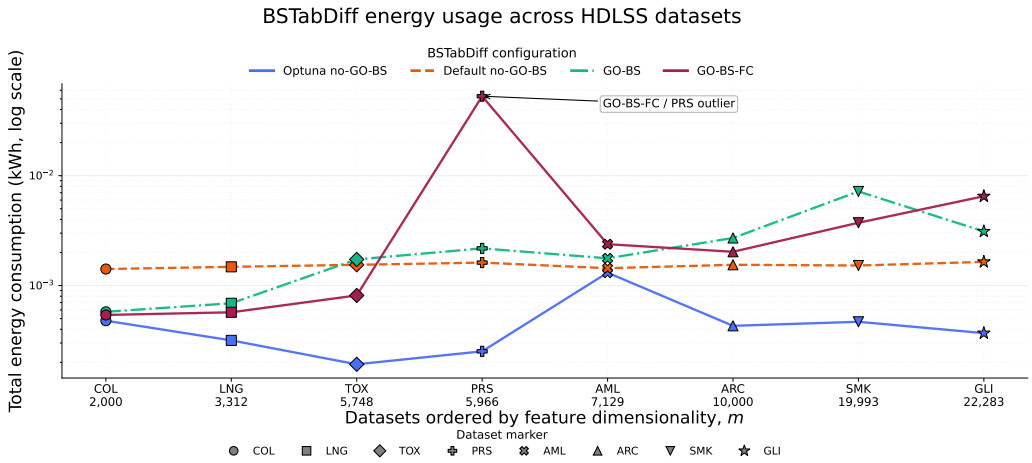
Pareto Trade-off Analysis for Resource Usage. Total energy usage is approximately linearly related to CO_2e emissions, while higher memory consumption and longer training time generally increase energy usage and, consequently, CO_2e emissions. We further analyze the resource-utility trade-off of top tabular generative models by comparing dataset dimensionality with CPU RSS and GPU memory usage, and by constructing Pareto frontiers over memory/time versus TSTR accuracy. For each generator g, we compute the mean TSTR accuracy $\bar{A}_g$, mean training time $\bar{T}_g$, and mean total memory $\overline{\mathrm{Mem}}_g$ across the $D = 8$ HDLSS datasets, where we get Eq. 55.

$$\overline{\mathrm{Mem}}_g = \frac{1}{D}\sum_{d=1}^{D}\left(\mathrm{Mem}_{g,d}^{\mathrm{CPU}} + \mathrm{Mem}_{g,d}^{\mathrm{GPU}}\right) \tag{55}$$

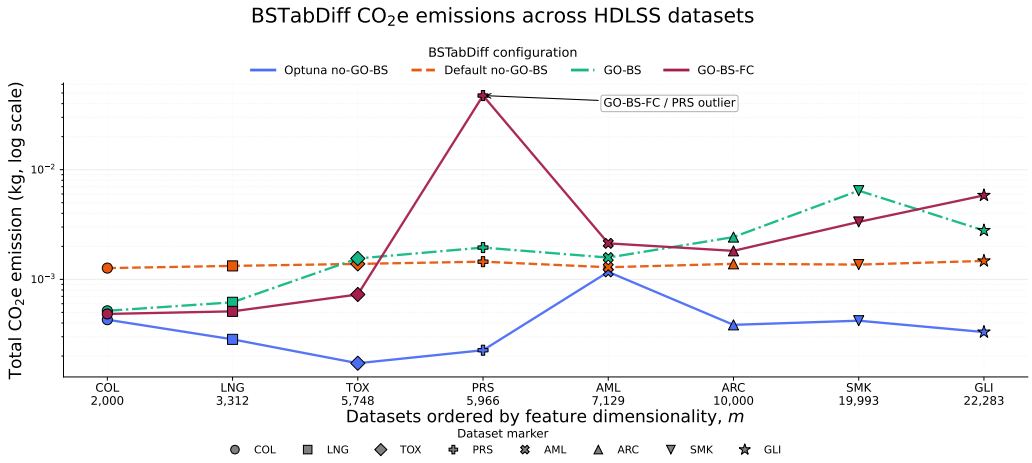
A generator is Pareto-efficient if no other generator achieves both lower resource usage and higher TSTR accuracy, i.e., generator g_i is dominated by generator g_j when $R_{g_j} \leq R_{g_i}$ and $\bar{A}_{g_j} \geq \bar{A}_{g_i}$, with at least one strict inequality, where R_g denotes either $\overline{\mathrm{Mem}}_g$ or $\bar{T}_g$. As shown in Fig. A9.2 and Table A9.2, BSTabDiff is the only generator that remains Pareto-efficient under both resource criteria. It achieves the highest mean TSTR accuracy of 86.33% while requiring only 6.73 seconds of mean training time and 1.741 GiB of mean total memory. In contrast, several baselines obtain competitive TSTR accuracy but require substantially larger GPU memory or training time; for example, TabDiff reaches 81.91% mean TSTR accuracy but uses 41.535 GiB mean total memory, while TVAE and CTGAN require several thousand seconds on average. These results indicate that BSTabDiff provides a stronger accuracy-efficiency balance, especially in HDLSS settings where scaling with feature dimensionality and memory footprint is critical.

A10 Pseudocode

Algorithms A10.1 and A10.2 describe the generation and training procedures of BSTabDiff, respectively. Algorithm A10.2 first determines the canonical feature order and block partition using GO-BS or GO-BS-FC, and estimates empirical inverse marginal CDFs from observed training values, optionally conditioned on class labels. It then infers block latents $h \in \mathbb{R}^M$ through pooled rank-Gaussianization and standardized within-block averages, fits continuous emissions by least squares and categorical emissions by cross-entropy on observed entries, and estimates missingness probabilities from training masks. Finally, it learns the compact prior $p_\theta(h \mid Y)$ using conditional maximum likelihood for flows or denoising score matching for diffusion. Algorithm A10.1 synthesizes samples by optionally drawing Y, sampling $h \sim p_\theta(h \mid Y)$, generating a canonical missingness



(a) Energy consumption.



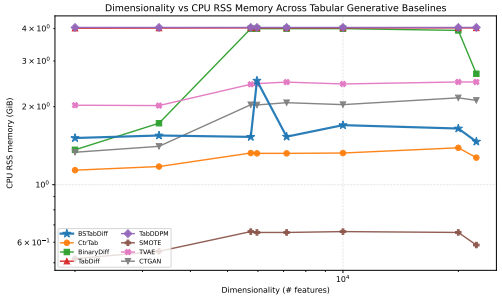
(b) CO_2e emissions.

Figure A9.1: Energy and carbon footprint of BSTabDiff variants across HDLSS datasets ordered by feature dimensionality. Curves denote configurations and marker shapes denote datasets. Both axes use log-scale resource measurements.

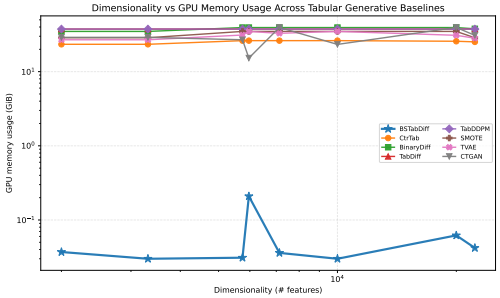
Table A9.2: Resource Pareto summary across HDLSS datasets. Mean TSTR accuracy is reported in percentage. Training time is measured in seconds, and memory usage is measured in GiB. Total memory is computed as CPU RSS memory plus GPU memory usage. Light green marks the best predictive performance, light blue marks the most efficient resource values, and bold Pareto indicators identify non-dominated models.

Model	Mean TSTR Acc. (%)	Mean Time (s)	Mean CPU RSS (GiB)	Mean GPU Mem. (GiB)	Mean Total Mem. (GiB)	#Datasets	Mem.-Pareto	Time-Pareto
BSTabDiff	**86.33**	**6.73**	1.682	**0.060**	**1.741**	8	**Yes**	**Yes**
CtrTab	45.77	102.29	1.283	25.304	26.588	8	No	No
BinaryDiff	71.37	488.55	3.212	38.017	41.229	8	No	No
TabDiff	81.91	79.67	4.023	37.512	41.535	8	No	No
TabDDPM	65.67	100.00	4.040	37.512	41.552	8	No	No
SMOTE	81.80	43.70	**0.618**	32.537	33.155	8	No	No
TVAE	80.38	4599.47	2.356	30.840	33.196	8	No	No
CTGAN	57.64	6914.84	1.899	29.150	31.049	8	No	No

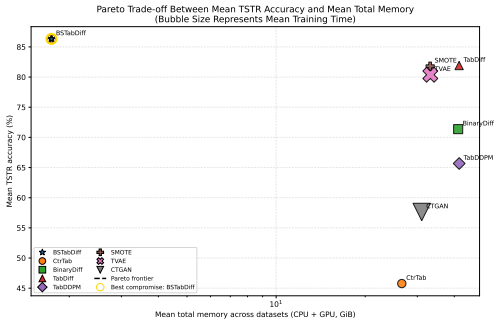
mask $\tilde{R}$ from empirical observation probabilities, and decoding each feature through its block subunit h_t using copula-Gaussian emissions and inverse marginals or categorical logits. The generated features and masks are restored to the original feature order using the fixed learned permutation $\hat{\pi}$. Algorithm A10.3 details GO-BS, which constructs a sparse feature-affinity graph, estimates a global feature ordering, initializes contiguous block boundaries by minimizing cross-block leakage, and refines the ordering and boundaries. GO-BS-FC (Algorithm A10.4) provides a feature-clustered alternative that clusters features in $X^\top$, constructs local feature orderings, orders the clusters using representative vectors, and concatenates the resulting sequences. The learned permutation and



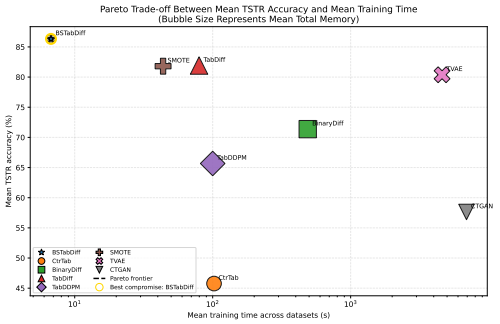
(a) Dimensionality vs. CPU RSS memory.



(b) Dimensionality vs. GPU memory usage.



(c) Pareto trade-off between mean total memory and mean TSTR accuracy.



(d) Pareto trade-off between mean training time and mean TSTR accuracy.

Figure A9.2: Resource-TSTR accuracy trade-offs across tabular generative baselines. Top row: dataset dimensionality versus CPU RSS memory and GPU memory usage. Bottom row: Pareto frontiers comparing mean TSTR accuracy against mean total memory and mean training time.

boundaries $(\hat{\pi}, \hat{\mathbf{b}})$ define the canonical block structure used by BSTabDiff, with GO-BS-FC offering a computationally efficient alternative for high-dimensional datasets.

A11 Additional Fidelity Diagnostics

BSTabDiff's fidelity diagnostics in Table A11.1 show that, on most HDLSS datasets, the generator produces synthetic distributions that are reasonably close to the real data while still avoiding trivial memorization. We first probe marginal fidelity via per-feature KS and W1 distances, which are modest on Colon and ALLAML and somewhat larger on Lung and (especially in scale) GLI-85, indicating that BSTabDiff captures many univariate marginals but struggles more on heavily skewed, high-variance features. Pairwise structure is assessed through Pearson and Spearman $|\Delta\mathrm{corr}|$, which remain in a moderate regime (mean $\approx$ 0.17–0.22) across datasets, suggesting that the generator preserves a substantial fraction of the real correlation graph rather than collapsing to independent noise. Higher-order moments (HOM: mean, variance, skewness, kurtosis) further reveal that Colon, ALLAML, and Lung have relatively small average discrepancies, whereas GLI-85 exhibits inflated variance- and mean-scale differences, consistent with its extreme HDLSS regime and raw feature scaling. Label-conditional structure is evaluated via $|\Delta\mathrm{MI}(\text{feature}, y)|$, where Colon and ALLAML show moderate MI gaps, and Lung has a larger mean MI discrepancy, implying that, for Lung, synthetic samples only coarsely approximate the most discriminative features. We also monitor NaN rates, which are essentially zero in the real data and remain at $\mathcal{O}(10^{-4})$ in the synthetic tables, and a nearest neighbor privacy diagnostic (Priv.), where large NN distances between synthetic and real points indicate that the generator does not simply copy training samples. Finally, the C2ST accuracy/AUC quantifies how easily a classifier can distinguish real from synthetic data: for Colon, ALLAML, and GLI-85, C2ST performance is at or even below chance, indicating that simple discriminators do not find a stable separating signal across folds, whereas Lung exhibits a much stronger C2ST signal, highlighting it as the most challenging dataset where BSTabDiff leaves a clearer "synthetic" footprint despite still providing useful downstream TSTR performance. Table A11.2 reports fidelity diagnostics for BSTabDiff under the Optuna-tuned hyperparameter setting without feature permutation. Overall, the results indicate that BSTabDiff preserves key statistical structure reasonably well even in challenging HDLSS regimes. In particular, Colon and ALLAML

Algorithm A10.1 Block-Subunit HDLSS Tabular Generation

Require: Canonical block partition $\{\mathcal{S}_t\}_{t=1}^{M}$, optional class label or label sampler $p(Y)$, learned prior $p_\theta(h \mid Y)$, empirical observation probabilities $\{\hat{\rho}_j(Y)\}$, inverse continuous marginals $\{\hat{F}_{j,Y}^{-1}\}$, categorical emission parameters $\{W_j, c_j(Y)\}$ (if applicable), and fixed permutation $\hat{\pi}$ (identity if ordering is disabled)
Ensure: Synthetic sample (X, R, Y) in the original feature order

Select label Y ▷ sample or specify; omit if unconditional
Sample block latents $h = (h_1, \ldots, h_M) \sim p_\theta(h \mid Y)$ ▷ diffusion or flow prior
Initialize canonical vectors $\tilde{X} \leftarrow (\mathrm{NA}, \ldots, \mathrm{NA})$ and $\tilde{R} \leftarrow \mathbf{0}_m$
for $t = 1$ to M **do**
  for each feature $j \in \mathcal{S}_t$ **do**
    Sample observation indicator $\tilde{R}_j \sim \mathrm{Bernoulli}\big(\hat{\rho}_j(Y)\big)$
    if $\tilde{R}_j = 0$ **then**
      $\tilde{X}_j \leftarrow \mathrm{NA}$
    else
      if j is continuous **then**
        Sample copula-Gaussian latent $Z_j \leftarrow a_j h_t + b_j(Y) + \xi_j$, $\xi_j \sim \mathcal{N}(0, \hat{\sigma}_j^2)$
        Map to uniform $U_j \leftarrow \Phi(Z_j)$
        Apply inverse marginal $\tilde{X}_j \leftarrow \hat{F}_{j,Y}^{-1}(U_j)$
      else ▷ categorical / discrete
        Compute logits $\ell_j \leftarrow W_j h_t + c_j(Y)$
        Sample $\tilde{X}_j \sim \mathrm{Categorical}\big(\mathrm{softmax}(\ell_j)\big)$
      end if
    end if
  end for
end for
Restore original feature order: $X \leftarrow \tilde{X}_{\hat{\pi}}$, $R \leftarrow \tilde{R}_{\hat{\pi}}$
return (X, R, Y)

show consistently small marginal discrepancies, moderate correlation errors, and stable higher-order moment matching, suggesting that the generator captures both first-order and dependency structure well. Although Lung remains the most difficult case, the model still maintains finite Wasserstein and rank-correlation discrepancies with negligible NaN rates, indicating stable generation despite the stronger label-related complexity. GLI exhibits large scale-sensitive Wasserstein, moment, and privacy values, which is expected given its extreme dimensionality and feature magnitude, but its relatively small KS and correlation discrepancies still suggest that the synthetic data preserves substantial distributional structure at the normalized relational level. Across all four datasets, the near-zero NaN fractions further confirm that BSTabDiff remains numerically robust under the tuned no-permutation configuration.

A12 Additional Ablation Studies

Table A12.1 and Fig. A12.1 report compact ablations of BSTabDiff on Colon evaluated with TabPFN-2.5. Overall, BSTabDiff remains robust: most variants achieve high downstream utility (TSTR AUC $\approx$ 0.87-0.90) and consistently track the real data upper bound (TRTR AUC $\approx$ 0.93), yielding strong efficiency ratios (TSTR/TRTR $\approx$ 0.94-0.96). The best TSTR is obtained with more synthetic samples (`nSyn500`), while changing the latent prior (FlowPrior), removing EMA, or varying the number of blocks ($M = 16/64$) has only minor impact on utility. In contrast, the discriminator signal (C2ST AUC) is more sensitive to these choices, suggesting that some settings produce synthetic data that is easier to distinguish even when predictive utility remains comparable.

A12.1 Selecting GO-BS versus GO-BS-FC

Shape and scale-aware motivation. GO-BS and GO-BS-FC provide complementary mechanisms for constructing dependency-aware feature blocks. GO-BS first clusters samples to estimate local feature-affinity structures and aggregates their orderings, whereas GO-BS-FC clusters features directly

Algorithm A10.2 Training BSTabDiff HDLSS Generator (diffusion or flow prior)

Require: Training dataset $\mathcal{D} = \{(X^{(i)}, R^{(i)}, Y^{(i)})\}_{i=1}^{n}$ (labels optional), feature types, block budget M, ordering configuration, and prior family $p_\theta(h \mid Y)$ (diffusion or flow)
Ensure: Fitted prior θ, emission parameters ϕ, empirical marginals, observation probabilities, canonical blocks $\{\mathcal{S}_t\}_{t=1}^{M}$, and permutation $\hat{\pi}$

if GO-BS or GO-BS-FC is enabled **then**
  Estimate feature ordering and boundaries $(\hat{\pi}, \hat{\mathbf{b}})$ using the selected GO-BS variant
  Construct contiguous canonical blocks $\{\mathcal{S}_t\}_{t=1}^{M}$
else
  Set $\hat{\pi} \leftarrow \mathrm{id}$ and use supplied or equally divided blocks
end if
Reorder features, masks, and feature specifications into canonical space: $X^{\dagger} \leftarrow X_{\hat{\pi}^{-1}}$, $R^{\dagger} \leftarrow R_{\hat{\pi}^{-1}}$
Fit empirical continuous marginals $\{\hat{F}_{j,Y}\}$ using observed training values ▷ class-conditional if enabled
Retain interpolated inverse marginals $\{\hat{F}^{-1}_{j,Y}\}$ for generation
Rank-Gaussianize observed continuous features: $Z_{ij} \leftarrow \Phi^{-1}\left(\frac{r_{ij}-1/2}{N_j}\right)$ ▷ r_{ij} is the observed rank; N_j is the observed count
Use pooled ranks across classes; set unobserved Gaussianized entries to zero
for each training sample i and block t **do**
  Infer the scalar block factor $\hat{h}_{it} \leftarrow \frac{\sum_{j \in \mathcal{S}_t^{\mathrm{cont}}} R^{\dagger}_{ij} Z_{ij}}{\max\left(1, \sum_{j \in \mathcal{S}_t^{\mathrm{cont}}} R^{\dagger}_{ij}\right)}$
end for
Standardize each column of the inferred latent matrix $\hat{H} \in \mathbb{R}^{n \times M}$
Estimate empirical observation probabilities $\hat{\rho}_j(Y)$ from training masks ▷ class-conditional if available
Fit continuous emission coefficients $a_j, b_j(Y)$ and residual variances $\hat{\sigma}_j^2$ by least squares on observed Gaussianized entries
Fit categorical emission parameters $W_j, c_j(Y)$ by cross-entropy on observed categories
for each prior-training minibatch (h, Y) from $(\hat{H}, Y)$ **do**
  if flow prior on h **then**
    Update θ by maximizing $\log p_\theta(h \mid Y)$ ▷ conditional MLE
  else ▷ diffusion prior on h
    Sample timestep τ and noise ϵ
    Form noisy latent h_τ via $q(h_\tau \mid h)$
    Update θ to minimize $\|\epsilon - \epsilon_\theta(h_\tau, \tau, Y)\|_2^2$ ▷ score matching
  end if
end for
return $(\theta, \phi, \hat{\pi}, \{\mathcal{S}_t\}, \{\hat{F}_{j,Y}, \hat{F}^{-1}_{j,Y}\}, \{\hat{\rho}_j(Y)\})$

and performs ordering within smaller feature groups. Selecting between these approaches requires considering both the statistical characteristics and computational scale of a dataset. The feature-to-sample ratio $\rho = m/n$, used in DynaTab (Habib et al., 2026c) and GOTabPFN (Habib et al., 2026a), characterizes dataset shape but does not alone capture total matrix scale, since datasets with similar ρ can contain substantially different numbers of data cells. Motivated by BeyondArena (Purucker et al., 2026), we therefore consider both ρ and $\kappa = nm$. We reuse the original regime assignment $\mathrm{Reg}(D)$ and the Micro-Macro refinement introduced in Appendix A13.2. Specifically, Eq. 59 defines the regime-specific median threshold T_r over a fixed reference collection, and Eq. 60 assigns each dataset to Micro or Macro relative to the threshold of its original regime. The resulting classification is illustrated in Fig. A13.2.

Default variant-selection rule. Let $\mathcal{D}_{\mathrm{FC}}$ denote the set of datasets assigned to Macro-HDLSS or Macro-HDHSS under the Micro-Macro refinement:

Algorithm A10.3 GO-BS: Graph-guided Ordering with Block Segmentation

Require: Training data $X \in \mathbb{R}^{n \times m}$, number of blocks M, affinity metric ϕ, top-k sparsification parameter k, local window size w, refinement passes P
Ensure: Estimated permutation $\hat{\pi}$ and boundary vector $\hat{\mathbf{b}}$
Normalize or standardize feature columns of X
Build sparse feature-affinity graph $G = (V, E, W)$ using metric ϕ
Retain top-k neighbors per feature and symmetrize the graph
Compute a coarse ordering $\pi^{(0)} \leftarrow \big(\mathrm{SpectralOrder}(G)\big)^{-1}$ ▷ feature-to-position map
Compute initial boundaries:
$\mathbf{b}^{(0)} \leftarrow \mathrm{SegmentByLeakage}\big(\pi^{(0)}, G, M\big)$
Initialize accepted solution: $\hat{\pi} \leftarrow \pi^{(0)}$, $\hat{\mathbf{b}} \leftarrow \mathbf{b}^{(0)}$
for $p = 1$ to P **do**
    $\pi' \leftarrow \mathrm{LocalReorder}\big(\hat{\pi}, \hat{\mathbf{b}}, G, w\big)$
    $\mathbf{b}' \leftarrow \mathrm{BoundaryRefine}\big(\pi', \hat{\mathbf{b}}, G, M, w\big)$
    if $\mathcal{J}_{\text{GO-BS}}\big(\pi', \mathbf{b}'\big) \geq \mathcal{J}_{\text{GO-BS}}\big(\hat{\pi}, \hat{\mathbf{b}}\big)$ **then**
        break
    end if
    $\hat{\pi} \leftarrow \pi'$
    $\hat{\mathbf{b}} \leftarrow \mathbf{b}'$
end for
return $(\hat{\pi}, \hat{\mathbf{b}})$

$$\mathcal{D}_{\mathrm{FC}} = \big\{ D : \mathrm{Scale}(D) = \mathrm{Macro}, \quad \mathrm{Reg}(D) \in \{\mathrm{HDLSS}, \mathrm{HDHSS}\} \big\}$$

We recommend the following default ordering-variant selection rule:

$$\mathcal{V}(D) = \begin{cases} \text{GO-BS-FC}, & D \in \mathcal{D}_{\mathrm{FC}} \\ \text{GO-BS}, & \text{otherwise} \end{cases} \tag{56}$$

Accordingly, GO-BS-FC is recommended for Macro-HDLSS and Macro-HDHSS datasets, where high feature dimensionality and large data matrix scale may make global feature ordering computationally expensive. Feature clustering decomposes the ordering task into smaller within-cluster problems while preserving the block organization required by BSTabDiff. GO-BS remains the default for Micro-HDLSS, Micro-HDHSS, and other regimes, where sample-clustered feature-affinity estimation is computationally manageable and can capture heterogeneous dependency patterns across sample subpopulations. This rule is a practical, regime-aware heuristic rather than a guarantee that either variant is universally optimal.

Empirical support and computational efficiency. The computational ablation in Table A3.2 shows that GO-BS-FC is faster than GO-BS on seven of the eight initial HDLSS datasets. Notable reductions include 151.93 to 60.61 seconds on SMK and 62.24 to 24.42 seconds on ARC. These savings are accompanied by competitive downstream utility, as reported in Table A3.1, supporting the use of feature clustering when ordering cost becomes substantial. Conversely, the sample clustered GO-BS formulation remains useful when feature dependencies vary across local sample populations, since its cluster-specific affinity graphs explicitly represent such heterogeneity.

Comparison with fixed global graph ordering. Appendix A6 provides a complementary evaluation of ordering solvers on a fixed global feature-affinity graph. Table A6.1 compares downstream TSTR performance across greedy, Christofides, simulated annealing, threshold accepting, genetic algorithm, and ant colony ordering procedures, with GO-BS and GO-BS-FC included as structural reference methods. Table A6.2 additionally reports their ordering and training costs. In the GO-BS versus GO-BS-FC reference comparison on Colon, GO-BS-FC achieves TSTR accuracy of 87.15% versus 86.82% and TSTR AUC of 90.73% versus 90.30%, while reducing ordering time from 23.57 to 1.08 seconds (approximately $21.8\times$) and total training time from 28.33 to 15.60 seconds. These

Algorithm A10.4 GO-BS-FC: Feature-Clustered Graph-guided Ordering with Block Segmentation

Require: Training data $X \in \mathbb{R}^{n \times m}$, number of feature clusters K_f, number of blocks M, affinity metric ϕ, top-k sparsification parameter k, refinement passes P
Ensure: Estimated permutation $\hat{\pi}$ and boundary vector $\hat{\mathbf{b}}$
Transpose the data to obtain feature-as-sample representation $X^\top \in \mathbb{R}^{m \times n}$
Normalize or standardize feature vectors in $X^\top$
Cluster features into K_f groups:
$\{\mathcal{F}_1, \ldots, \mathcal{F}_{K_f}\} \leftarrow \mathrm{FeatureCluster}(X^\top, K_f)$
for $c = 1$ to K_f **do**
    Build sparse within-cluster affinity graph $G_c = (V_c, E_c, W_c)$ over features in $\mathcal{F}_c$ using metric ϕ
    Retain top-k neighbors per feature and symmetrize G_c
    Compute local feature order $\mathbf{o}_c \leftarrow \mathrm{GraphOrder}(G_c, P)$
    Compute cluster representative $r_c \leftarrow \mathrm{Aggregate}(X_{:,\mathbf{o}_c})$
end for
Build cluster-level affinity graph G_{clust} over representatives $\{r_c\}_{c=1}^{K_f}$
Compute cluster ordering $\boldsymbol{\zeta} \leftarrow \mathrm{GraphOrder}(G_{\mathrm{clust}}, P)$
Concatenate local feature orders according to $\boldsymbol{\zeta}$:
$\mathbf{o} \leftarrow \mathbf{o}_{\zeta_1} \| \mathbf{o}_{\zeta_2} \| \cdots \| \mathbf{o}_{\zeta_{K_f}}$
Define $\hat{\pi}(\mathbf{o}_r) \leftarrow r$ for $r = 1, \ldots, m$ ▷ feature-to-position map
Build global sparse feature-affinity graph $G_{\mathrm{feat}} = (V, E, W)$ using metric ϕ
Compute block boundaries:
$\hat{\mathbf{b}} \leftarrow \mathrm{SegmentByLeakage}\big(\hat{\pi}, G_{\mathrm{feat}}, M\big)$
return $(\hat{\pi}, \hat{\mathbf{b}})$

reference results are reported separately from the identical protocol fixed global graph solver comparison and therefore should not be interpreted as additional solvers evaluated under that same protocol.

Near-threshold selection and stability diagnostics. For datasets with κ close to the regime-specific threshold T_r, the default rule may be supplemented with cluster stability diagnostics under bootstrap resampling. More stable sample clusters favor GO-BS, whereas more stable feature clusters favor GO-BS-FC, particularly when computational costs are comparable. This provides a secondary qualitative diagnostic rather than a separately validated selection criterion. The Micro-Macro thresholds should be estimated from a fixed reference collection before applying the rule to new datasets, thereby avoiding dataset-dependent recalibration during evaluation. Overall, the proposed strategy retains GO-BS as the canonical sample conditioned ordering formulation while recommending GO-BS-FC as a computationally efficient default for Macro-HDLSS and Macro-HDHSS settings.

A13 HDLSS Literature and Datasets Used

A13.1 Dataset Categorization Rules.

There is no universally accepted numerical threshold that uniquely defines a dataset as HDLSS. In the HDLSS literature, the term is commonly used for regimes where the ambient dimension, or number of variables, is much larger than the sample size. Following the empirical regime rules introduced by Habib et al. (2026c,a), we use the feature-to-sample ratio $\rho = m/n$ to develop operational decision boundaries in the (n, m) space (Fig. A13.1a). This is often formalized through HDLSS asymptotics (Aoshima et al., 2018), where the dimension grows while the sample size is fixed or grows much more slowly (Hall et al., 2005; Jung and Marron, 2009). Thus, the simple condition "$m > n$" provides a useful heuristic, but it is too coarse to describe the full practical range of high-dimensional regimes. To make the categorization more operational, we empirically stratify datasets using the feature-to-sample ratio $\rho = m/n$, which allows us to distinguish qualitatively different settings beyond a binary HDLSS versus non-HDLSS label. Let n be the number of samples, m be the number of features, and $\rho = \frac{m}{n}$ be the feature-to-sample ratio. We can assign each dataset

Table A11.1: Fidelity diagnostics for BSTabDiff synthetic data on four HDLSS datasets under the default configuration. We report marginal distances (KS, W1), correlation discrepancies, higher-order moment (HOM) gaps, label-feature MI gaps, NaN rates, nearest neighbor privacy distance, and C2ST mean±std over 5×5 CV. Here, KS = Kolmogorov-Smirnov, W1 = Wasserstein-1, MI = mutual information, and Priv. = synthetic-to-real nearest neighbor distance.

Metric	Statistic	COL	AML	LNG	GLI
KS (per feature)	mean	0.0848	0.1047	0.2001	0.1592
	max	0.2387	0.2917	0.6650	0.4471
W1	mean	0.1252	0.1563	0.1036	3.76×10^{2}
	max	0.3910	0.8026	0.6745	1.63×10^{4}
Pearson $\lvert\Delta\mathrm{corr}\rvert$	mean	0.2183	0.1874	0.2072	0.1695
	max	1.1046	1.2400	0.9875	1.3613
Spearman $\lvert\Delta\mathrm{corr}\rvert$	mean	0.2179	0.1893	0.2110	0.1705
	max	1.0889	1.1925	1.0696	1.2419
HOM	mean	7.99×10^{-2}	8.94×10^{-2}	7.88×10^{-2}	3.19×10^{2}
	var	1.55×10^{-1}	2.53×10^{-1}	5.08×10^{-2}	2.40×10^{6}
	skew	0.1666	0.4664	0.4580	0.5397
	kurt	0.4819	2.3330	2.2799	2.8462
$\lvert\Delta\mathrm{MI}($**feature**$, y)\rvert$	mean	0.2218	0.1656	0.6274	0.1661
	max	0.5115	0.5435	1.1457	0.5458
NaN fraction	real	0.0000	0.0000	0.0000	0.0000
	synthetic	1.10×10^{-4}	1.03×10^{-4}	1.13×10^{-4}	1.02×10^{-4}
Priv.	mean	4.97×10^{1}	9.21×10^{1}	1.53×10^{1}	4.67×10^{5}
	min	4.68×10^{1}	8.42×10^{1}	1.20×10^{1}	4.27×10^{5}
	p1	4.72×10^{1}	8.43×10^{1}	1.21×10^{1}	4.28×10^{5}
	p5	4.77×10^{1}	8.54×10^{1}	1.23×10^{1}	4.36×10^{5}
C2ST ACC	mean ± std	0.2880 ± 0.1023	0.2600 ± 0.1149	0.7401 ± 0.0437	0.2286 ± 0.1059
C2ST AUC	mean ± std	0.1614 ± 0.0795	0.1536 ± 0.1285	0.7029 ± 0.0610	0.1312 ± 0.0954

to one of five regimes according to the following empirical ρ-based criteria according to the rules defined in Table A13.1. Fig. A13.1a visualizes the empirical regime decision boundaries in the (n, m) space, highlighting how the HDLSS, HDHSS, LDHSS, LDLSS, and MixedRegime categories depend jointly on n, m, and $\rho = m/n$.

A13.2 Micro-Macro Dataset Regime Refinement

Motivation. The dataset categorization rules used by Habib et al. (2026c,a) characterize datasets through the number of samples n, the number of features m, and the feature-to-sample ratio $\rho = m/n$. These criteria define five empirical regimes: HDLSS, HDHSS, LDHSS, LDLSS, and MixedRegime. Complementarily, BeyondArena (Purucker et al., 2026) considers dataset scale in terms of the number of data cells. Motivated by this perspective, we define the shape-scale parameters by Eq. 57.

$$\rho = \frac{m}{n}, \qquad \kappa = nm \tag{57}$$

While ρ characterizes the relative proportions of features and samples, κ measures the total number of entries in the data matrix, providing a complementary indicator of dataset size.

Regime-specific scale thresholds. The typical magnitude of κ can differ substantially across the original regimes. Consequently, a single global threshold would not provide a regime-normalized distinction between small and large datasets. Let us assume we get Eq. 58 which denotes the set of original regimes, and let $\mathrm{Reg}(D) \in \mathcal{C}$ denote the original regime assignment of dataset D according to Table A13.1.

$$\mathcal{C} = \{\mathrm{HDLSS}, \mathrm{HDHSS}, \mathrm{LDHSS}, \mathrm{LDLSS}, \mathrm{MixedRegime}\} \tag{58}$$

Consider a fixed reference benchmark collection $\mathcal{B}_{\mathrm{ref}} = \{D_i\}_{i=1}^{N}$, where dataset D_i contains n_i samples and m_i features, with $\kappa_i = n_i m_i$. For each regime $r \in \mathcal{C}$ represented in the reference collection, define the regime-specific median scale threshold by Eq. 59. The reference collection and resulting thresholds are fixed before assigning Micro-Macro labels.

$$T_r = \mathrm{median}\,\{\kappa_i : D_i \in \mathcal{B}_{\mathrm{ref}}, \mathrm{Reg}(D_i) = r\} \tag{59}$$

Table A11.2: Fidelity diagnostics for BSTabDiff synthetic data under Optuna-tuned hyperparameters without feature permutation. We report marginal distances (KS, W1), correlation discrepancies, higher-order moment (HOM) gaps, label-feature MI gaps, NaN rates, and synthetic-to-real nearest neighbor privacy distance. Here, KS = Kolmogorov-Smirnov, W1 = Wasserstein-1, MI = mutual information, and Priv. = privacy distance.

Metric	Statistic	COL	AML	LNG	GLI
KS (per feature)	mean	0.1126	0.0945	0.1988	0.0904
	max	0.3326	0.3111	0.6527	0.2859
W1	mean	0.1753	0.1450	0.1108	1.98×10^{2}
	max	0.5513	0.7975	0.6013	1.11×10^{4}
Pearson $\lvert\Delta\mathrm{corr}\rvert$	mean	0.2178	0.1516	-	0.1460
	max	1.0592	1.2636	-	1.3425
Spearman $\lvert\Delta\mathrm{corr}\rvert$	mean	0.2167	0.1563	0.1767	0.1485
	max	1.0652	0.9608	0.9693	1.1352
HOM	mean	1.43×10^{-1}	1.08×10^{-1}	7.66×10^{-2}	1.48×10^{2}
	var	1.49×10^{-1}	2.00×10^{-1}	6.44×10^{-2}	1.61×10^{6}
	skew	0.2091	0.3620	0.3852	0.3474
	kurt	0.5005	1.9362	1.9097	2.0479
$\lvert\Delta\mathrm{MI}($**feature**$, y)\rvert$	mean	0.2228	0.2165	0.5730	0.2117
	max	0.4818	0.5274	1.0507	0.5161
NaN fraction	real	0.0000	0.0000	0.0000	0.0000
	synthetic	1.10×10^{-4}	1.11×10^{-4}	9.81×10^{-5}	9.83×10^{-5}
Priv.	mean	5.03×10^{1}	9.30×10^{1}	1.73×10^{1}	4.82×10^{5}
	min	4.69×10^{1}	7.79×10^{1}	1.28×10^{1}	4.33×10^{5}
	p1	4.75×10^{1}	7.91×10^{1}	1.33×10^{1}	4.35×10^{5}
	p5	4.84×10^{1}	8.04×10^{1}	1.36×10^{1}	4.41×10^{5}

Micro-Macro assignment. For a dataset D with scale $\kappa = nm$, we assign a Micro or Macro label relative to the reference median of its original regime by Eq. 60. The refined dataset regime is defined as Eq. 61. This yields 10 possible refined categories: Micro-HDLSS, Macro-HDLSS, Micro-HDHSS, Macro-HDHSS, Micro-LDHSS, Macro-LDHSS, Micro-LDLSS, Macro-LDLSS, Micro-MixedRegime, and Macro-MixedRegime.

$$\mathrm{Scale}(D) = \begin{cases} \text{Micro}, & \kappa < T_{\mathrm{Reg}(D)} \\ \text{Macro}, & \kappa \geq T_{\mathrm{Reg}(D)} \end{cases} \tag{60}$$

$$\mathrm{Reg}_{\mathrm{MM}}(D) = \big(\mathrm{Scale}(D), \mathrm{Reg}(D)\big) \tag{61}$$

Interpretation. This refinement preserves the original statistical regime definitions while introducing a relative measure of data-matrix scale within each regime. It distinguishes datasets with similar statistical shapes but substantially different numbers of data cells, enabling more granular, regime-stratified benchmark analysis. The refinement does not directly characterize algorithmic runtime or memory complexity, which also depend on the underlying model architecture and computational operations. Furthermore, the Micro-Macro distinction is relative to the chosen reference collection rather than a universal dataset-size boundary. Fig. A13.2 visualizes the refinement in the (n, m) space, where constant-scale boundaries satisfy $\kappa = nm = T_r$ within each original regime.

A13.3 HDLSS in Cross-Domain Settings.

The availability of HDLSS datasets varies substantially across domains, but the regime is most commonly observed in biomedical settings, particularly in omics, gene-expression, and microarray datasets. This pattern is also consistent with prior high-dimensional tabular learning studies: TabPFN-Wide (Kolberg et al., 2025) explicitly identifies HDLSS as a frequent scenario in biomedical applications, while ProtoGate (Jiang et al., 2024) evaluates on seven real-world biomedical HDLSS datasets. LSPIN/LLSPIN (Yang et al., 2022) considers six real-world high-dimensional

Table A12.1: Key results for BSTabDiff compact ablations on Colon using TabPFN-2.5. Values are the mean ± standard deviation over the evaluation folds. Efficiency is defined as the AUC ratio of TSTR/TRTR. Lower C2ST AUC indicates that real and synthetic data are harder to distinguish in this experiment.

Setting	TSTR AUC ↑	TRTR AUC ↑	C2ST AUC ↓	Eff. (TSTR/TRTR) ↑
BASE	0.886 ± 0.075	0.932 ± 0.072	0.101 ± 0.058	0.950
FlowPrior	0.896 ± 0.078	0.931 ± 0.070	0.180 ± 0.100	**0.963**
noEMA	0.886 ± 0.075	0.931 ± 0.072	0.192 ± 0.113	0.952
noCC-Marg	0.872 ± 0.098	0.930 ± 0.070	$\mathbf{0.070 \pm 0.030}$	0.938
noCC-Miss	0.889 ± 0.078	0.931 ± 0.070	0.190 ± 0.114	0.954
M16	0.886 ± 0.075	0.932 ± 0.072	0.188 ± 0.111	0.951
M64	0.896 ± 0.078	0.932 ± 0.072	0.205 ± 0.110	0.961
ep1500	0.896 ± 0.078	0.931 ± 0.070	0.236 ± 0.137	0.962
ep5000	0.893 ± 0.076	0.932 ± 0.072	0.170 ± 0.089	0.958
nSyn100	0.892 ± 0.078	0.931 ± 0.070	0.239 ± 0.117	0.958
nSyn500	$\mathbf{0.901 \pm 0.080}$	$\mathbf{0.941 \pm 0.075}$	0.223 ± 0.104	0.958

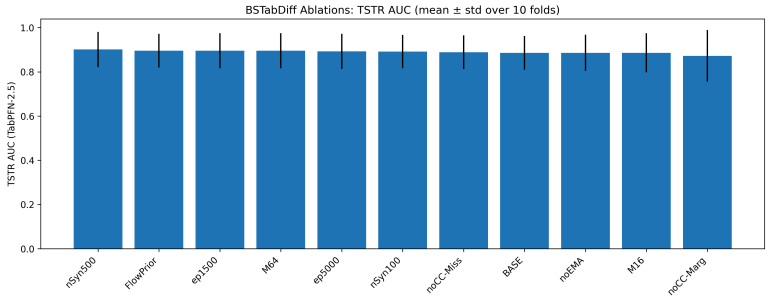
Table A13.1: Empirical dataset regime categorization rules based on the number of samples n, number of features m, and feature-to-sample ratio $\rho = m/n$ (Habib et al., 2026c,a).

Regime	Rule
HDLSS	$m > 1000,\ n < 1000,\ \rho > 2$
HDHSS	$m > 1000,\ n > 10^4,\ 0.005 < \rho \leq 2$
LDHSS	$m \leq 100,\ n > 10^4,\ \rho \leq 0.01$
LDLSS	$m \leq 100,\ n \leq 1000,\ \rho \leq 0.05$
MixedRegime	*otherwise*

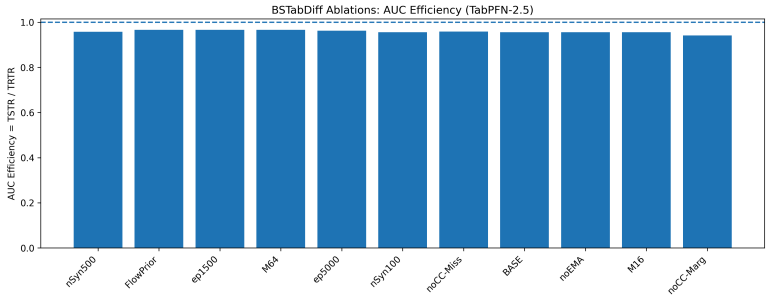
datasets spanning biomedical and text domains; however, under our proposed regime rules, its text datasets do not satisfy the HDLSS criteria and instead fall into the MixedRegime category. Similarly, TabPFN-Wide uses four real-world biomedical multi-omics HDLSS datasets, while BETA (Liu and Ye, 2025), another TabPFN variant capable of handling HDLSS data, evaluates on a broader collection of high-dimensional classification benchmarks covering biomedical, text, face-image, and other benchmark datasets. Under our categorization, BETA's benchmarks span HDLSS, HDHSS, and MixedRegime settings, but no explicit financial HDLSS dataset is included among its 20 real-world high-dimensional experiments. TuneTables (Feuer et al., 2024), another high-dimensionality compatible TabPFN variant, evaluates broadly on TabZilla (McElfresh et al., 2023) and large-scale tabular benchmarks, but likewise does not include high-dimensional financial datasets. For tabular generative modeling, CtrTab (Li et al., 2026) is one of the few works to include high-dimensional datasets; among its datasets, Arcene is the sole biomedical HDLSS dataset, whereas Stadyn, with 6248 samples and 1084 features, falls into MixedRegime under our rules in Table A13.1. Overall, related high-dimensional literature repeatedly relies on biomedical datasets, with some extensions to text, face-image, and sensor-like domains. This suggests that HDLSS is predominantly a biomedical phenomenon, with partial occurrence in text and image-derived feature domains, whereas financial datasets are not commonly represented in this regime. Fig. A13.1b summarizes the cross-domain coverage of high-dimensional datasets in related literature, highlighting the historical concentration of HDLSS benchmarks in biomedical applications and the broader evaluation scope of recent methods such as GOTabPFN (Habib et al., 2026a), which spans 16 datasets across biomedical, text, face/image, camera/biometrics, and image embedding domains. The figure also illustrates the varying dimensionality regimes represented across these domains.

A14 Representation Quality via t-SNE

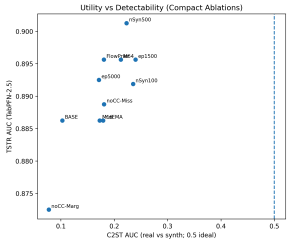
Fig. A14.1 provides a qualitative sample-space visualization on the Colon dataset. We jointly embed real and synthetic samples using t-SNE (Maaten and Hinton, 2008) after shared standardization and PCA (Hotelling, 1933) pre-reduction. The non-GO-BS variant generates samples that broadly follow



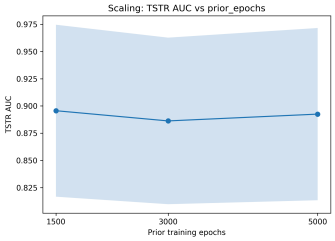
(a) TSTR AUC (compact ablations).



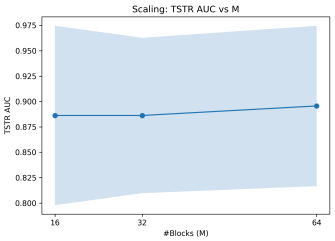
(b) AUC efficiency (TSTR/TRTR).



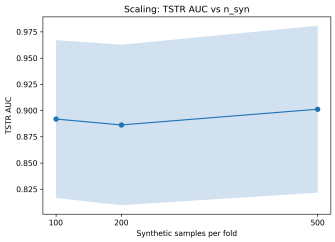
(c) Utility vs detectability (TSTR vs C2ST).



(d) Scaling: TSTR AUC vs prior epochs.



(e) Scaling: TSTR AUC vs #blocks M.



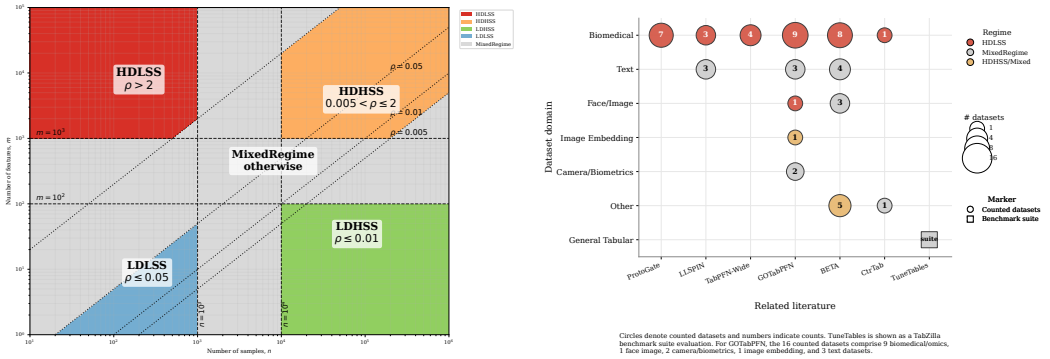
(f) Scaling: TSTR AUC vs n_{syn}.

Figure A12.1: BSTabDiff compact ablations (TabPFN-2.5). The first two rows show the main compact-ablation summaries, while the last two rows show utility-privacy tradeoff and scaling trends.

the real data geometry, but the class-conditional structure is less clearly separated. In contrast, the GO-BS variant produces synthetic samples that more closely align with the corresponding real class regions, particularly in the dominant class-specific clusters. This suggests that organizing features through GO-BS before block-subunit generation helps BSTabDiff preserve local dependency patterns and discriminative sample geometry in HDLSS tabular data.

A15 Statistical Significance Analysis

We assess statistical significance using TSTR accuracy from Logistic Regression across the eight HDLSS datasets, considering only synthetic generators; Real (TRTR) is retained only as an upper-bound reference. BSTabDiff obtains the highest average TSTR accuracy (86.33 ± 7.62) and ranks first on every dataset. A Friedman (Friedman, 1937) omnibus test over all synthetic generators



(a) Regime decision map in the (n, m) space.

(b) Cross-domain usage in related literature.

Figure A13.1: Overview of HDLSS and related high-dimensional regimes. (a) Dataset regimes in (n, m) space, with $\rho = m/n$ (Habib et al., 2026a,c); dashed lines mark n, m thresholds and dotted diagonals mark constant ρ. (b) Cross-domain coverage in prior HDLSS-related literature, highlighting the historical concentration in biomedical datasets and the expanding evaluation scope of recent methods such as GOTabPFN (Habib et al., 2026a), which covers 16 datasets spanning biomedical, text, face/image, camera/biometrics, and image embedding domains. Bubble sizes indicate dataset counts, while colors distinguish dimensionality regimes; benchmark suite evaluations are represented separately.

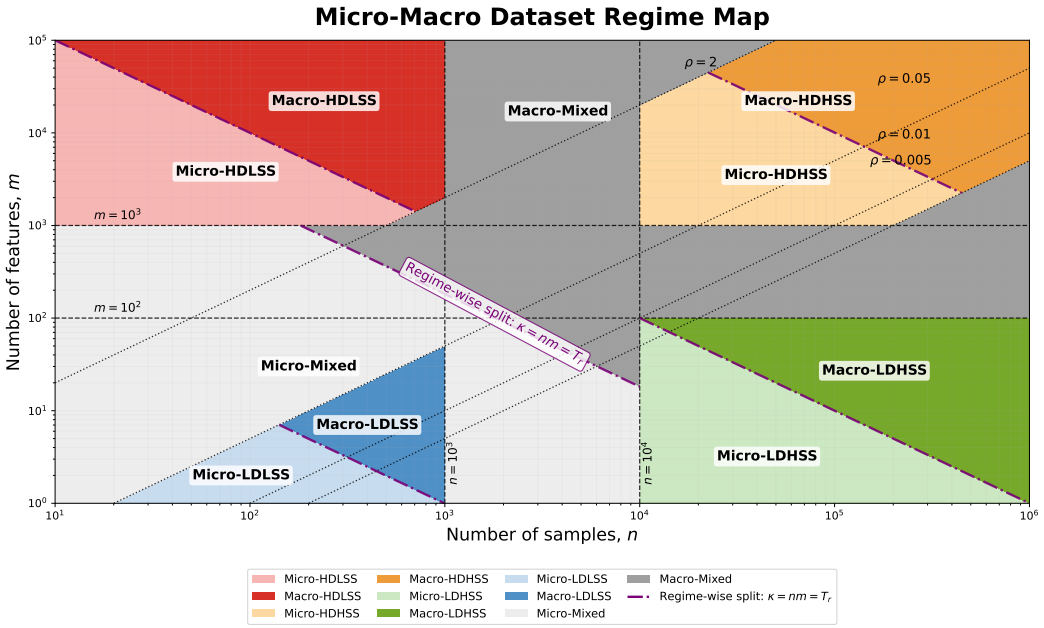


Figure A13.2: **Micro-Macro refinement of dataset regimes.** The five original regimes are partitioned using the shape parameter $\rho = m/n$ and scale parameter $\kappa = nm$. Purple dash-dotted curves indicate regime-specific median boundaries $\kappa = T_r$, separating Micro ($\kappa < T_r$) and Macro ($\kappa \geq T_r$) subsets.

indicates significant differences among methods ($\chi^2 = 75.43$, $p = 3.91 \times 10^{-12}$). Pairwise one-sided Wilcoxon signed-rank tests (Demšar, 2006) with Holm correction (Nemenyi, 1963) show that BSTabDiff improves over every baseline on all eight datasets, with corrected $p_{\mathrm{H}} = 0.0391$ in each comparison (Table A15.1).

A16 BSTabDiff Hyperparameters

Table A16.1 summarizes the dataset-specific block latent size M used across the eight HDLSS datasets. All other hyperparameters were kept fixed across datasets: we used a diffusion prior trained for 10,000 epochs with batch size 128, learning rate 10^{-3}, EMA enabled with decay 0.999, no feature permutation, and no predefined blocks. Thus, the main dataset-specific adjustment was the block latent size M, which was increased for higher-dimensional datasets (e.g., GLI and SMK)

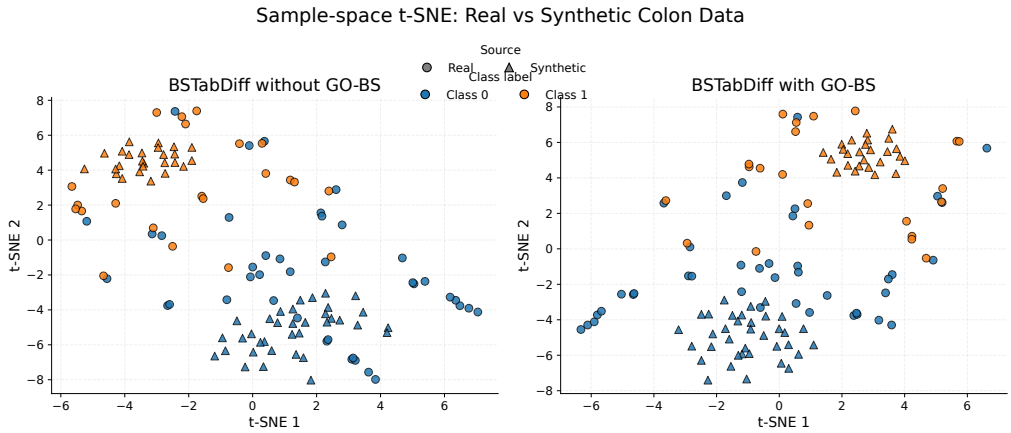


Figure A14.1: t-SNE of real and synthetic Colon samples from BSTabDiff with and without GO-BS. GO-BS yields better class-conditional alignment with the real manifold, suggesting improved preservation of HDLSS structure.

Table A15.1: Pairwise TSTR comparison of BSTabDiff against synthetic baselines on eight HDLSS datasets. ΔAcc. is the mean gain in percentage points; W/T/L denotes dataset-level wins/ties/losses; p_{H} is Holm-corrected. Light green shades emphasize larger gains, and light blue marks statistically significant Holm-corrected results.

Baseline	Δ	**W/T/L**	p	p_{H}	**Baseline**	Δ	**W/T/L**	p	p_{H}
CtrTab	**40.56**	**8/0/0**	0.0039	**0.0391**	BinaryDiff	**14.96**	**8/0/0**	0.0039	**0.0391**
CTGAN	**28.70**	**8/0/0**	0.0039	**0.0391**	ForestDiff	**12.77**	**8/0/0**	0.0039	**0.0391**
CTAB-GAN	**21.97**	**8/0/0**	0.0039	**0.0391**	TVAE	**5.95**	**8/0/0**	0.0039	**0.0391**
CTAB-GAN+	**21.15**	**8/0/0**	0.0039	**0.0391**	SMOTE	**4.53**	**8/0/0**	0.0039	**0.0391**
TabDDPM	**20.66**	**8/0/0**	0.0039	**0.0391**	TabDiff	**4.42**	**8/0/0**	0.0039	**0.0391**

and kept smaller for lower-dimensional ones (e.g., COL and TOX). Table A16.2 summarizes the best BSTabDiff hyperparameters selected by Optuna under the no GO-BS, no permutation setting. Across datasets, the tuning consistently preferred relatively small block counts ($M = 8$ in most cases, with $M = 16$ for PRS and TOX), while diffusion priors were favored on most datasets except COL and AML, where flow priors performed best. The selected learning rates and EMA decays varied substantially across datasets, indicating nontrivial dataset-specific optimization structure, whereas the synthetic sample budget used in the inner-loop proxy evaluation typically remained moderate (64–256). These results support using dataset-specific tuning following the tuning protocol set by Kotelnikov et al. (2023).

Table A16.3 summarizes the GO-BS-only hyperparameters selected by Optuna when tuning the ordering and block-segmentation module separately from the main BSTabDiff generator. Because GO-BS requires feature-affinity graph construction and graph-based refinement, whose cost scales quadratically with the number of features, we used a smaller 150-trial tuning budget for this component. The selected configurations vary across datasets, indicating that the optimal graph metric, sparsification level, number of sample clusters, and boundary-refinement strength are dataset dependent. In particular, correlation-based affinities are preferred on COL, SMK, AML, and TOX, while cosine, Manhattan, and KL-divergence are selected for other datasets, suggesting that GO-BS benefits from adapting its structural prior to the geometry of each HDLSS dataset.

A17 Additional Runtime and Resource Usage

Runtime and resources. Table A17.1 provides an additional runtime and memory profile for BSTabDiff, one fixed-configuration training run finishes in roughly one minute with very small peak GPU usage and modest CPU RSS, supporting the computational efficiency of the block-subunit formulation.

Assets and licenses. All datasets and baseline implementations are appropriately cited, with original sources provided in Tables A23.1 and A23.2. We respect their respective licenses and terms of use. The released code includes documentation, dependencies, and licensing information. New assets, including GitHub, PyPI, project webpages, and the Hugging Face demo, are linked in the NeurIPS checklist under "New assets."

Table A16.1: Dataset-specific block latent size M used for BSTabDiff across the 8 HDLSS datasets.

Dataset	COL	LNG	GLI	SMK	AML	PRS	ARC	TOX
M	32	64	128	192	64	64	64	32

Table A16.2: Best BSTabDiff hyperparameters from 2000 Optuna trials per HDLSS dataset, selected by Logistic Regression validation on synthetic data, with GO-BS and feature permutation disabled.

Data	M	Prior	Epochs	Batch	LR	EMA	n_{syn}
COL	8	flow	300	64	3.2931e-03	0.9926	64
LNG	8	diffusion	1200	128	5.8095e-05	0.9998	200
GLI	8	diffusion	800	32	3.1386e-04	0.9911	64
SMK	8	diffusion	1200	32	1.4595e-04	0.9995	256
AML	8	flow	1600	32	4.9452e-03	0.9928	128
PRS	16	diffusion	1200	64	2.5615e-04	0.9983	256
ARC	8	diffusion	1200	16	1.6479e-04	0.9990	200
TOX	16	diffusion	300	64	5.8193e-04	0.9991	64

A18 Generalization across Diverse Data Regimes

Generalization Beyond the Original HDLSS Scope. We extend BSTabDiff's evaluation to nine additional datasets spanning HDLSS, HDHSS, LDHSS, and MixedRegime settings, including TCGA-BRCA (omics), Springleaf (marketing), CIFAR-10 ResNet-50 embeddings, MiniBooNE, Adult, ORLraws10P (face images), and three high-dimensional bag-of-words text datasets (BASEHOCK, RELATHE, and PCMAC). These benchmarks cover heterogeneous data characteristics, including mixed continuous and categorical features, feature-wise missingness, and substantially different sample-to-feature ratios. The six datasets with additional accuracy evaluations range from 100 to 130,064 samples and from 14 to 10,304 features. As reported in Table 4 of the main paper, at least one BSTabDiff variant achieves the highest synthetic data TSTR AUC on each of the nine datasets, with BSTabDiff-no-p obtaining the best average AUC of 89.21%, compared with 85.16% for the strongest non-BSTabDiff baseline (SMOTE). Complementary TSTR accuracy results in Table A18.1 show a consistent pattern across the six datasets with accuracy evaluation: at least one BSTabDiff variant attains the highest synthetic data TSTR accuracy on every dataset. BSTabDiff-no-p achieves the best six dataset average TSTR accuracy of 80.19%, followed closely by BSTabDiff-d (80.05%) and BSTabDiff-BS (80.01%), all surpassing the strongest non-BSTabDiff baseline, SMOTE (79.04%), while remaining below the real data TRTR reference (85.36%). These results provide additional evidence that the block-subunit design remains effective beyond the original HDLSS setting.

Effect of Feature Ordering on synthetic data Fidelity. To investigate the role of feature ordering beyond downstream predictive utility, we compare four BSTabDiff configurations on CIFAR-10 ResNet-50 embeddings. Although their TSTR AUC values are nearly identical (98.37-98.40%; Table 4), Table A18.2 reveals complementary fidelity characteristics. Tuned BSTabDiff-FC achieves the lowest Kolmogorov-Smirnov (KS), W1, mean, and variance discrepancies, whereas tuned BSTabDiff-BS achieves the lowest Pearson and Spearman correlation gaps. The tuned no-permutation configuration obtains the smallest mutual-information discrepancy. Privacy distances remain comparable across configurations, and all variants exhibit negligible synthetic NaN fractions. These findings indicate

Table A16.3: Best GO-BS-only hyperparameters from Optuna tuning per HDLSS dataset, with feature permutation disabled. GO-BS is tuned separately due to the approximately $O(m^2)$ cost of feature-affinity graph operations. Ref. Ord./Pass = local order refinement/passes; Dir. Sel. = direction selection; Bnd. Pass/Win. = boundary refinement passes/window; λ_{cross} = cross-block leakage penalty; γ_{bal} = block-balance penalty.

Data	Clusters	Metric	Bins	Top-k	Ref. Ord.	Dir. Sel.	Ref. Pass	Bnd. Pass	Bnd. Win.	λ_{cross}	γ_{bal}
COL	3	correlation	24	16	False	False	3	5	16	2.3989	5.19e-06
GLI	3	manhattan	32	32	True	False	1	0	24	2.4373	9.27e-06
LNG	3	cosine	32	16	True	False	1	5	16	1.1691	2.65e-08
PRS	3	kl_divergence	24	16	True	True	0	5	12	0.1892	3.78e-05
SMK	7	correlation	64	64	True	True	2	5	12	0.3032	2.68e-07
TOX	4	correlation	48	64	False	False	1	3	16	2.3373	3.25e-08
AML	3	correlation	16	16	False	True	2	8	8	0.1607	1.86e-05
ARC	7	cosine	48	32	True	False	2	5	12	2.7504	7.58e-04

Table A17.1: BSTabDiff runtime/resource usage and energy footprint measured on Vulcan under the default no-GO-BS configuration. Peak GPU is max CUDA memory; CPU memory is end-of-run RSS. Energy is total consumption (kWh) and CO_2e is estimated carbon emission (kg).

Dataset	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)	Dataset	Time (s)	GPU (GiB)	CPU (GiB)	Energy (kWh)	CO_2e (kg)
COL	49.20	0.025	2.752	1.414e-3	1.267e-3	**SMK**	59.61	0.043	1.284	1.522e-3	1.364e-3
GLI	63.32	0.044	1.269	1.643e-3	1.473e-3	**TOX**	49.71	0.032	1.243	1.545e-3	1.384e-3
LNG	53.05	0.031	1.223	1.481e-3	1.327e-3	**AML**	42.67	0.028	1.220	1.439e-3	1.290e-3
PRS	51.63	0.027	1.219	1.618e-3	1.450e-3	**ARC**	52.12	0.032	1.257	1.546e-3	1.385e-3

Table A18.1: **Extended TSTR accuracy (%).** Results on six additional datasets. Gold, silver, and bronze cells indicate the first-, second-, and third-highest distinct synthetic data results, respectively; ties share the same color. Bold values denote the best synthetic data result for each dataset. TRTR is the real data reference and is excluded from ranking. Avg(n) denotes the mean over the six available datasets.

Model	BRCA*,†,‡	CIFAR^{+}	MB$^{\#}$	Spring$^{+,\dagger,\ddagger}$	ORL*	Adult$^{\#}$	Avg(n)
BSTabDiff-no-p	81.16 ± 3.13	**82.42 ± 0.47**	85.62 ± 2.94	65.82 ± 1.27	**86.80 ± 5.64**	79.30 ± 1.59	**80.19 (6)**
BSTabDiff-BS	**81.59 ± 2.93**	82.28 ± 0.44	83.76 ± 11.32	**66.04 ± 1.23**	**86.80 ± 5.64**	79.61 ± 1.09	80.01 (6)
BSTabDiff-FC	81.36 ± 2.67	82.25 ± 0.48	85.76 ± 2.51	64.73 ± 1.91	86.00 ± 6.48	79.48 ± 1.32	79.93 (6)
BSTabDiff-d	81.11 ± 3.29	82.22 ± 0.68	**86.73 ± 1.07**	63.72 ± 1.58	86.60 ± 6.28	**79.93 ± 1.10**	80.05 (6)
TabDiff	81.25 ± 3.97	71.89 ± 2.65	82.27 ± 1.18	64.94 ± 5.55	44.20 ± 10.28	76.22 ± 2.16	70.13 (6)
CtrTab	81.40 ± 3.12	76.77 ± 1.58	81.51 ± 1.60	65.12 ± 2.60	82.40 ± 5.80	75.76 ± 2.12	77.16 (6)
TabDDPM	81.42 ± 0.29	66.10 ± 2.78	76.12 ± 6.96	65.74 ± 0.01	60.10 ± 6.01	73.69 ± 5.46	70.53 (6)
BinaryDiff	53.99 ± 18.29	52.10 ± 2.36	54.84 ± 22.08	53.54 ± 12.69	51.20 ± 9.27	76.94 ± 2.39	57.10 (6)
SMOTE	81.49 ± 3.32	79.83 ± 1.46	84.21 ± 2.55	65.30 ± 2.94	85.40 ± 5.58	77.98 ± 1.53	79.04 (6)
TRTR	90.41 ± 1.74	87.45 ± 0.54	87.45 ± 0.24	68.76 ± 0.61	98.40 ± 2.73	79.67 ± 0.64	85.36 (6)

Dataset notation: BRCA = TCGA-BRCA, CIFAR = CIFAR-10 ResNet-50 embeddings, MB = MiniBooNE, Spring = Springleaf, ORL = ORLraws10P. *HDLSS; $^{+}$HDHSS; $^{\#}$LDHSS; †mixed continuous and categorical features; ‡feature-wise missing values.

Ranking highlights: 1st , 2nd , 3rd . Ranks are determined from the reported dataset-level mean accuracies and exclude TRTR.

that GO-BS and GO-BS-FC can improve different aspects of synthetic data fidelity even when downstream TSTR performance changes only minimally, highlighting the importance of evaluating marginal, dependency, and moment preservation alongside predictive utility.

A19 Block-Latent Dimension Selection and Sensitivity Analysis

Motivation and notation. The number of blocks M is a central hyperparameter of BSTabDiff, controlling both the dimensionality of the block-latent representation $h \in \mathbb{R}^M$ and the granularity of the feature partition. An excessively small M may limit the resolution of the learned block structure, whereas a large M can increase the computational burden of dependency-aware block construction. We therefore distinguish between a practical initialization rule and subsequent dataset-specific hyperparameter selection. Consistent with Appendix A13.2, let n denote the number of samples and m the number of input features. We use the shape parameter $\rho = m/n$ and the scale parameter $\kappa = nm$, where ρ characterizes feature-to-sample imbalance and κ represents the total number of data-matrix cells. The block count satisfies $1 \leq M \leq m$.

Practical selection of M. For tuned configurations, Optuna (Akiba et al., 2019) selects dataset-specific values of M together with other hyperparameters. For the default configuration and as practical guidance for previously unseen datasets, we introduce a two stage initialization based on ρ and κ. This strategy is motivated by the complexity-adaptive NSC design of GOTabPFN (Habib et al., 2026a), which avoids unnecessary compression for low-dimensional inputs and adjusts representation complexity according to dataset characteristics. Considering sample size and feature dimensionality jointly, rather than relying on m alone, is also consistent with BeyondArena's dataset-scale characterization (Purucker et al., 2026). Specifically, low-dimensional datasets with $m \leq 64$ retain one block per feature. For datasets with $m > 64$ but $\rho < 10$, we use a nominal budget $M = 64$. For strongly high-dimensional datasets with $m > 64$ and $\rho \geq 10$, the initial budget increases according to the matrix scale κ:

Table A18.2: **CIFAR-10 ResNet-50 fidelity across BSTabDiff variants.** Gold, silver, and bronze cells indicate the first-, second-, and third-best distinct values for each metric, respectively; ties share the same color. Bold denotes the best result for each metric. Lower is better for all discrepancy metrics and NaN fraction; higher privacy distance indicates less direct sample proximity.

Method	KS ↓	W1 ↓	Pearson ↓	Spearman ↓	Mean gap ↓	Variance gap ↓	$\lvert\Delta\text{MI}\rvert$ ↓	NaN fraction ↓	Privacy mean ↑
BSTabDiff-no-p	0.0810	0.0556	0.0966	0.1009	0.0314	0.0672	**0.2322**	$\mathbf{8.79 \times 10^{-5}}$	17.733
BSTabDiff-BS	0.0817	0.0563	**0.0965**	**0.1008**	0.0319	0.0666	0.2323	$\mathbf{8.79 \times 10^{-5}}$	17.698
BSTabDiff-d	0.0915	0.0630	0.1009	0.1064	0.0436	0.0657	0.2375	1.03×10^{-4}	**17.812**
BSTabDiff-FC	**0.0773**	**0.0538**	0.0967	0.1015	**0.0306**	**0.0655**	0.2350	$\mathbf{8.79 \times 10^{-5}}$	17.613

Configurations: no-p: tuned without permutation; BS: tuned with GO-BS; FC: tuned with GO-BS-FC; d: default configuration.

Metrics: KS, W1, correlation, moment, and mutual-information values represent mean feature-wise discrepancies. Privacy mean is a sample-distance proxy, not a formal privacy guarantee.

Ranking highlights: 1st , 2nd , 3rd . The three tuned configurations share the lowest reported NaN fraction.

$$M_{\text{init}} = \begin{cases} m, & m \le 64 \\ 64, & m > 64,\ \rho < 10 \\ \min\{M_0(\kappa), m\}, & m > 64,\ \rho \ge 10 \end{cases} \tag{62}$$

where

$$M_0(\kappa) = \begin{cases} 32, & \kappa < 2.5 \times 10^5 \\ 64, & 2.5 \times 10^5 \le \kappa < 10^6 \\ 128, & 10^6 \le \kappa < 3 \times 10^6 \\ 192, & \kappa \ge 3 \times 10^6 \end{cases} \tag{63}$$

The condition $\rho \ge 10$ is used only to identify strong feature-to sample imbalance for block-budget initialization; it does not redefine the five dataset regimes in Table A13.1. Likewise, the thresholds in Eq. 63 are fixed heuristic budget thresholds, distinct from the regime-specific Micro-Macro median thresholds T_r in Appendix A13.2. Consequently, M_{init} is a practical starting point rather than a theoretically optimal block count.

Dataset-wise application. Table A19.1 summarizes the application of the proposed (ρ, κ)-based block-budget selection rule across the eight original HDLSS benchmarks and nine additional datasets evaluated later. The rule reproduces the reported default budgets for COL, LNG, AML, PRS, GLI, and SMK, while ARC and TOX retain smaller values because their block-coherence or validation-utility diagnostics stabilized at more conservative budgets. For the additional datasets, MiniBooNE and Adult retain $M = m$ because $m \le 64$, whereas TCGA-BRCA, Springleaf, CIFAR-10, BASEHOCK, RELATHE, and PCMAC use the nominal budget $M = 64$ because they do not satisfy the strong-HDLSS initialization condition. The extreme-HDLSS ORLraws10P dataset receives $M = 128$. For consistency with the original benchmark metadata, ρ and κ are computed from the full dataset dimensions rather than individual training-fold sizes. These results illustrate how the proposed rule adapts the default block budget across diverse dimensionality regimes while allowing conservative adjustments when smaller values provide satisfactory validation utility. The proposed selection rule provides a simple initialization strategy across different dataset shapes and scales. In particular, datasets with $m \le 64$ retain one block per feature, datasets with $m > 64$ and $\rho < 10$ use a nominal budget of 64, and strongly HDLSS datasets use a scale-dependent budget determined by κ. Although the default rule reproduces most reported settings, the ARC and TOX exceptions illustrate that practitioners may retain smaller values when block coherence or downstream validation utility stabilizes. Thus, the rule should be interpreted as a practical initialization rather than a universally optimal choice of M.

Completed sensitivity analysis. To evaluate the sensitivity of BSTabDiff to the block-latent dimension, we conduct experiments on the Colon dataset for $M \in \{8, 16, 32, 64, 128\}$, both with and without GO-BS. The experiments use Logistic Regression for downstream TSTR evaluation and strict 5×5 repeated cross-validation. All other generator settings are fixed at their tuned values within each configuration, and every reported row contains $25/25$ completed folds. Table A19.2 presents the

Table A19.1: **Dataset-wise application of the (ρ, κ)-based block-budget selection rule.** Here, n and m denote the number of samples and features, respectively, with $\rho = m/n$ and $\kappa = nm$. M_{init} denotes the heuristic initialization, while M_{default} denotes the block budget used in the default configuration. Peach cells highlight cases where a smaller budget was retained. All values are based on full dataset dimensions.

Dataset	Regime	n	m	ρ	κ	M_{init}	M_{default}
Original HDLSS benchmarks							
COL	HDLSS	62	2,000	32.26	124,000	32	32
LNG	HDLSS	203	3,312	16.32	672,336	64	64
AML	HDLSS	72	7,129	99.01	513,288	64	64
PRS	HDLSS	102	5,966	58.49	608,532	64	64
GLI	HDLSS	85	22,283	262.15	1,894,055	128	128
SMK	HDLSS	187	19,993	106.91	3,738,691	192	192
ARC	HDLSS	200	10,000	50.00	2,000,000	128	**64**
TOX	HDLSS	171	5,748	33.61	982,908	64	**32**
Additional datasets evaluated							
TCGA-BRCA	HDLSS	705	1,940	2.75	1,367,700	64	64
CIFAR-10	HDHSS	11,000	2,048	0.186	22,528,000	64	64
MiniBooNE	LDHSS	130,064	50	0.00038	6,503,200	50	50
Springleaf	HDHSS	29,046	1,876	0.065	54,490,296	64	64
ORLraws10P	HDLSS	100	10,304	103.04	1,030,400	128	128
Adult	LDHSS	48,842	14	0.00029	683,788	14	14
BASEHOCK	MixedRegime	1,993	4,862	2.44	9,689,966	64	64
RELATHE	MixedRegime	1,427	4,322	3.03	6,167,494	64	64
PCMAC	MixedRegime	1,943	3,289	1.69	6,390,527	64	64

Notes: The original eight datasets follow the block-budget settings reported in Table A16.1. For ARC and TOX, the default budgets were reduced relative to the heuristic initialization. The nine additional datasets include mixed-type omics and marketing data, image embeddings, particle physics, census data, face images, and high-dimensional bag of words text representations. The reported budgets refer to the default selection procedure and do not represent independently Optuna-tuned block counts.

Table A19.2: **Completed sensitivity analysis of block-latent dimension M on Colon.** All configurations are evaluated using strict 5×5 repeated cross-validation with $25/25$ completed folds. All other generator hyperparameters are fixed within each configuration. TSTR accuracy and AUC are reported as mean with subscript standard deviation. Time is the mean fitting time per fold.

Configuration	M	TSTR Acc. (%)	TSTR AUC (%)	Time/Fold (s)
Tuned GO-BS	8	$84.18_{\pm 10.69}$	$88.55_{\pm 9.45}$	11.76
Tuned GO-BS	16	$84.18_{\pm 10.69}$	$88.22_{\pm 9.83}$	36.02
Tuned GO-BS	32	$84.18_{\pm 10.69}$	$88.40_{\pm 9.98}$	179.09
Tuned GO-BS	64	$84.16_{\pm 10.46}$	$88.67_{\pm 9.52}$	1090.90
Tuned GO-BS	128	$83.87_{\pm 10.56}$	$88.35_{\pm 9.77}$	**7574.60**
Tuned no-GO-BS	8	$84.18_{\pm 10.69}$	$88.62_{\pm 9.76}$	5.13
Tuned no-GO-BS	16	$83.26_{\pm 10.95}$	$88.32_{\pm 9.52}$	5.89
Tuned no-GO-BS	32	$84.18_{\pm 10.69}$	$88.53_{\pm 9.95}$	6.20
Tuned no-GO-BS	64	$84.18_{\pm 10.69}$	$88.45_{\pm 9.72}$	6.13
Tuned no-GO-BS	128	$83.87_{\pm 10.56}$	$88.35_{\pm 9.39}$	5.97

Runtime highlights: Mint identifies the lowest fit time within each configuration; peach indicates the most expensive GO-BS setting. All reported means are based on the full 25 completed folds.

completed results, which supersede the preliminary partial-fold sensitivity analysis reported during additional experiments.

Sensitivity findings. The completed results indicate that downstream utility is relatively stable over a substantial variation in M. With GO-BS, mean TSTR AUC ranges from 88.22% to 88.67%, a difference of only 0.45 percentage points. Without GO-BS, the corresponding range is 88.32%-88.62%, a difference of 0.30 percentage points. Mean accuracy varies by at most 0.31 and 0.92 percentage points with and without GO-BS, respectively.These differences are small relative to the reported cross-validation variability and provide no consistent evidence that larger M improves downstream predictive performance. In contrast, fitting time increases substantially with M when GO-BS is enabled. The mean time rises from 11.76 s/fold at $M = 8$ to 36.02 s/fold at $M = 16$,

179.09 s/fold at $M = 32$, 1090.90 s/fold at $M = 64$, and 7574.60 s/fold at $M = 128$. This corresponds to an approximately $644\times$ increase from $M = 8$ to $M = 128$. By comparison, the no-GO-BS configuration remains near 5-6 s/fold throughout the investigated range. Thus, increasing M provides no consistent TSTR utility gain in this experiment while substantially increasing the computational cost of the GO-BS-enabled configuration. The results motivate selecting a sufficiently small block budget rather than increasing the latent resolution without additional validation evidence.

Practical recommendation. For previously unseen datasets, we recommend using the (ρ, κ)-based rule in Eqs. 62 and 63 to initialize the block budget. The practitioner may then refine the value through dataset-specific validation, considering both downstream predictive utility and block-coherence diagnostics. When multiple candidate budgets yield comparable performance, the smallest satisfactory value of M should be preferred, particularly when GO-BS is used. The Colon sensitivity experiment supports this conservative strategy: relatively small block budgets achieve comparable downstream performance at substantially lower computational cost. However, the observed stability is an empirical finding for the evaluated dataset and settings, not a guarantee that the same range of M is optimal for all datasets.

A20 Necessity of the Block-Subunit Structure

Motivation and parameter efficiency. A dense latent-factor model provides a natural alternative to the block-subunit formulation of BSTabDiff. For an observed feature dimension m and latent dimension $M \ll m$, a dense factor decoder of the form $Z = Wh + b(Y) + \xi$ uses a loading matrix $W \in \mathbb{R}^{m\times M}$, requiring $O(mM)$ loading coefficients and exhibiting the usual rotational ambiguity of factor models. In contrast, BSTabDiff assigns each observed feature to a single block latent and estimates one scalar loading per feature, reducing the emission-side loading complexity to $O(m)$. Importantly, the joint prior $p_\theta(h \mid Y)$ continues to model dependence among all M block latents, so the blocks are not assumed mutually independent. The block-subunit formulation therefore provides a parameter-efficient and interpretable inductive bias for HDLSS generation rather than imposing a universally necessary modeling assumption.

Relationship to structured latent-factor learning. The motivation for dependency-aware blocks is consistent with the use of sparse factor models (Carvalho et al., 2008), graphical models (Meinshausen and Bühlmann, 2006), latent trees (Choi et al., 2011), and spectral biclustering (Kluger et al., 2003) to represent structured dependencies in high-dimensional data. BSTabDiff differs from a dense latent-factor decoder by explicitly associating each latent coordinate with a dependency-coherent feature block. GO-BS estimates this organization from training data rather than assuming a meaningful original column order. Although the exact GO-BS optimization problem is NP-hard (Theorem A2.4), the implementation uses a sparse approximate solver rather than exact combinatorial optimization. Related evidence from GOTabPFN (Habib et al., 2026a) further motivates locality-aware feature ordering and compact subunit representations for HDLSS datasets, particularly when meaningful feature-block structure is present.

Matched experimental design. To examine whether the block-subunit structure provides benefits beyond generic latent dimensionality reduction, we evaluate three configurations on the Colon dataset ($n = 62$, $m = 2000$): (i) DenseFactor-PCA, which uses a dense latent-factor decoder; (ii) BlockSparse no-GO-BS, which imposes block-sparse loadings without learned feature ordering; and (iii) BSTabDiff w/ GO-BS, which additionally learns the dependency-aware block organization. All configurations use the same reduced latent dimension $M = 8$, flow prior, marginal-transformation model, missingness model, synthetic data budget, and downstream evaluation protocol. TSTR performance is evaluated using Logistic Regression under strict 5×5 repeated cross-validation. The comparison isolates the effects of decoder sparsity and learned block organization while keeping the remaining generative components matched.

Downstream utility. Table A20.1 presents the matched downstream evaluation. DenseFactor-PCA achieves the highest mean TSTR accuracy (85.62%), whereas the block-sparse configurations achieve higher mean TSTR AUC. Specifically, the dense baseline obtains 88.40% AUC, which increases to 89.28% with block-sparse loadings without GO-BS and to 89.93% when the block organization is learned through GO-BS. The corresponding reported AUC efficiencies are 1.0184, 1.0304, and 1.0380, respectively. Thus, BSTabDiff w/ GO-BS improves mean TSTR AUC by 1.53 percentage points over DenseFactor-PCA while using substantially fewer decoder loading parameters.

Table A20.1: **Matched dense-factor versus block-subunit TSTR evaluation on Colon dataset.** All configurations use $m = 2000$, $M = 8$, and strict 5×5 cross-validation with Logistic Regression. AUC efficiency measures TSTR AUC relative to the TRTR AUC reference. Values are reported as mean with subscript standard deviation. Gold highlights the best result for each varying utility metric.

Method	M	TSTR Acc.	TSTR AUC	TRTR Acc.	TRTR AUC	AUC Eff.
DenseFactor-PCA	8	$\mathbf{0.8562}_{\pm 0.0817}$	$0.8840_{\pm 0.1029}$	$0.8431_{\pm 0.1058}$	$0.8703_{\pm 0.1114}$	$1.0184_{\pm 0.0504}$
BlockSparse no-GO-BS	8	$0.8331_{\pm 0.1157}$	$0.8928_{\pm 0.1014}$	$0.8431_{\pm 0.1058}$	$0.8703_{\pm 0.1114}$	$1.0304_{\pm 0.0782}$
BSTabDiff w/ GO-BS	8	$0.8362_{\pm 0.1270}$	$\mathbf{0.8993}_{\pm 0.1030}$	$0.8431_{\pm 0.1058}$	$0.8703_{\pm 0.1114}$	$\mathbf{1.0380}_{\pm 0.0830}$

Notes: Higher TSTR accuracy, TSTR AUC, and AUC efficiency are better. TRTR values are identical across the matched configurations and are included as real data references. Gold and silver denote the highest and second highest varying results, respectively.

Table A20.2: **C2ST and computational evaluation over the 25 Colon dataset folds.** C2ST separability is defined as $\max(\text{AUC}, 1 - \text{AUC})$, with values closer to 0.5 preferred. Raw C2ST accuracy and AUC are retained as measured; separability accounts for classifier label orientation. Fit time and peak GPU usage are reported as mean with subscript standard deviation. Gold highlights the best separability, fit time, and memory values.

Method	C2ST Acc.	Raw AUC	C2ST Sep.	Fit Time (s)	GPU (GiB)
DenseFactor-PCA	$0.2160_{\pm 0.0746}$	$0.1438_{\pm 0.0637}$	$\mathbf{0.8562}_{\pm 0.0637}$	$6.02_{\pm 0.80}$	$0.029795_{\pm 0.000000}$
BlockSparse no-GO-BS	$0.2693_{\pm 0.0933}$	$0.1333_{\pm 0.0551}$	$0.8667_{\pm 0.0551}$	$\mathbf{5.89}_{\pm 0.88}$	$\mathbf{0.029744}_{\pm 0.000000}$
BSTabDiff w/ GO-BS	$0.2707_{\pm 0.0735}$	$0.1196_{\pm 0.0652}$	$0.8804_{\pm 0.0652}$	$10.48_{\pm 0.80}$	$0.076296_{\pm 0.000034}$

Notes: Lower C2ST separability, fit time, and peak GPU usage are preferred. Raw C2ST AUC values below 0.5 are converted to separability using $\max(\text{AUC}, 1 - \text{AUC})$. The reported statistics are based on the matched 5×5 evaluation.

Nevertheless, these differences are small relative to the cross-validation variability, and the dense baseline retains the highest TSTR accuracy. The results therefore support a metric dependent advantage of block-subunit factorization rather than universal superiority.

Marginal fidelity and real-synthetic distinguishability. The full data synthetic sample evaluation provides additional evidence concerning distributional fidelity. DenseFactor-PCA obtains mean KS and W1 discrepancies of 0.0964 and 0.1486, respectively. Imposing block-sparse loadings without GO-BS reduces these values to 0.0797 and 0.1127, while BSTabDiff w/ GO-BS achieves comparable discrepancies of 0.0799 and 0.1132. Thus, both block-sparse variants improve marginal fidelity relative to the dense baseline, although GO-BS does not further reduce KS or W1 compared with the block-sparse configuration without ordering. Table A20.2 reports complementary Classifier 2 Sample Test (C2ST) results over the 25 cross-validation folds. DenseFactor-PCA achieves the lowest C2ST separability (85.62%), compared with 86.67% for BlockSparse no-GO-BS and 88.04% for BSTabDiff w/ GO-BS. Since C2ST separability closer to 50% indicates greater real-synthetic indistinguishability, this comparison demonstrates that improvements in downstream AUC and marginal fidelity do not necessarily coincide with improvements in C2ST.

Parameter efficiency and computational cost. Table A20.3 compares the parameter counts and computational resources measured in the full data experiments. DenseFactor-PCA uses $mM = 16{,}000$ loading coefficients, whereas both block-sparse variants use only $m = 2{,}000$, yielding an eightfold reduction in decoder loading complexity. All three configurations share a flow prior containing $531{,}680$ parameters, resulting in total reported parameter counts of $547{,}680$ for DenseFactor-PCA and $533{,}680$ for the block-sparse variants. Thus, the eightfold reduction applies specifically to decoder loadings rather than to the entire generative model. BlockSparse no-GO-BS achieves the lowest full data training time (6.001 s) and peak GPU usage (0.0297 GiB), whereas DenseFactor-PCA achieves the lowest CPU resident memory (1.162 GiB). Enabling GO-BS increases the mean cross-validation fit time from 5.89 to 10.48 s/fold and the full data training time from 6.001 to 7.810 s relative to BlockSparse no-GO-BS, reflecting the additional cost of constructing a dependency-aware feature organization.

Relationship to block-structure assumptions. The block-subunit formulation is motivated by distinct within-block dependencies and sparse cross-block interactions (Assumption 3.2), but these conditions are not strict requirements for fitting BSTabDiff. We formalize the meaning of sparse latent dependence, provide practical diagnostics, and discuss assumption violations and possible remedies in Appendix A2.1. The preceding experiment evaluates the empirical benefit of block-subunit factorization but does not imply that the underlying dependence assumptions hold universally.

Table A20.3: **Full data resource and parameter comparison on Colon dataset.** The raw dimension is $m = 2000$ and the latent dimension is $M = 8$. DenseFactor-PCA uses $mM = 16{,}000$ loading coefficients, while the block-sparse models require $m = 2{,}000$. All configurations share the same flow prior parameter count. Gold highlights the lowest parameter and resource values, with ties sharing the same color.

Method	M	Loading Params.	Prior Params.	Total Params.	Train Time (s)	GPU (GiB)	CPU RSS (GiB)
DenseFactor-PCA	8	16,000	531,680	547,680	6.164	0.0298	**1.162**
BlockSparse no-GO-BS	8	**2,000**	531,680	**533,680**	**6.001**	**0.0297**	1.260
BSTabDiff w/ GO-BS	8	**2,000**	531,680	**533,680**	7.810	0.0765	1.476

Notes: Parameter counts follow the matched experimental accounting of decoder loadings and the common flow prior. Full data training times are distinct from mean cross-validation fit times in Table A20.2. GPU denotes peak GPU memory, and CPU RSS denotes resident CPU memory.

Assumption 3.2 introduces distinct within-block dependence and sparse cross-block interactions as sufficient structural conditions motivating the identifiability and sample complexity arguments. Such assumptions should be assessed through within-block affinity, normalized cross-block leakage, latent conditional dependence sparsity, and bootstrap stability of the recovered feature partitions. When these diagnostics indicate weak block structure or dense cross-block dependence, the benefits of block factorization may diminish. In these settings, practitioners may relax the cross-block penalty, reconsider the block budget M, or disable GO-BS when it provides no validation benefit. The joint prior $p_\theta(h \mid Y)$ nevertheless continues to model dependencies among the block latents, so the model does not require strict cross-block independence.

Interpretation and limitations. The controlled Colon ablation supports block-subunit factorization as a parameter-efficient, interpretable, and potentially variance-reducing inductive bias for HDLSS generation. Relative to DenseFactor-PCA, BSTabDiff w/ GO-BS achieves a higher mean TSTR AUC (89.93% versus 88.40%), higher AUC efficiency (1.0380 versus 1.0184), and lower KS and W1 discrepancies while using $2{,}000$ rather than $16{,}000$ decoder loadings. However, the dense baseline achieves the highest TSTR accuracy and the lowest C2ST separability, while the block-sparse configuration without GO-BS attains slightly better KS and W1 values and lower computational cost than BSTabDiff w/ GO-BS. Consequently, these results do not establish that block structure is universally necessary or superior for all data distributions and metrics. Rather, they demonstrate that explicitly constrained and learned feature blocks can provide useful gains in selected utility and fidelity measures while substantially reducing emission-side loading complexity.

A21 Component Sensitivity Analysis and Practical Configuration Guide

A21.1 "One Factor at a Time" Component Sensitivity

Experimental protocol. To assess the sensitivity of BSTabDiff to individual modeling and optimization choices, we conduct an additional "one factor at a time" analysis on the Colon dataset ($n = 62$, $m = 2000$). We evaluate tuned configurations with and without GO-BS using Logistic Regression under strict 5×5 repeated cross-validation. Each experiment varies one component while holding the remaining tuned generator parameters fixed. The investigated factors include the latent prior family, prior training epochs, prior learning rate, exponential moving average (EMA), and synthetic sample budget n_{syn}. We report TSTR accuracy, TSTR AUC, raw C2ST AUC, and mean fitting time in Table A21.1. The sensitivity of the block count M is evaluated separately in Appendix A19.

Overall sensitivity. Completed experiments demonstrate that downstream TSTR AUC is comparatively stable across investigated settings. With GO-BS, mean TSTR AUC ranges from 88.10% to 89.03%, compared w/ tuned baseline of 88.55%. Without GO-BS, corresponding range is 88.22%-89.10%, compared w/ baseline of 88.62%. The largest absolute change from respective tuned baseline is 0.0048 in both configurations, equivalent to 0.48 percentage points. These differences are small relative to reported cross-validation variability. Consequently, none of investigated prior or optimization settings produces a consistent large improvement in downstream AUC within tested ranges.

Component-wise changes. Table A21.2 summarizes the largest absolute change in mean TSTR AUC relative to the tuned baseline for each component. The prior family change produces the largest observed deviation (0.0048) in both configurations. Varying prior training epochs changes AUC by

Table A21.1: **Full component sensitivity analysis on Colon dataset.** Results use strict 5×5 repeated cross-validation with Logistic Regression. Each setting changes one component while holding the remaining tuned parameters fixed within each GO-BS configuration. Values are reported as mean with subscript standard deviation. C2ST AUC is the raw classifier AUC as reported in the experiment, without conversion to orientation-independent separability. Fit time is in seconds per fold.

Variant	Setting	TSTR Acc.	TSTR AUC	C2ST AUC	Fit Time (s)
GO-BS	**Base**	$0.8418_{\pm 0.1069}$	$0.8855_{\pm 0.0945}$	$0.1948_{\pm 0.1192}$	11.758
GO-BS	Matched diffusion prior	$0.8326_{\pm 0.1049}$	$0.8903_{\pm 0.0914}$	$0.2740_{\pm 0.1305}$	9.395
GO-BS	Prior epochs $= 150$	$0.8321_{\pm 0.1179}$	$0.8825_{\pm 0.0968}$	$0.2340_{\pm 0.1239}$	8.400
GO-BS	Prior epochs $= 800$	$0.8423_{\pm 0.1158}$	$0.8853_{\pm 0.0992}$	$0.2124_{\pm 0.1117}$	21.754
GO-BS	Prior epochs $= 1500$	$0.8356_{\pm 0.1065}$	$0.8832_{\pm 0.0991}$	$0.1804_{\pm 0.1167}$	35.037
GO-BS	Prior LR $= 0.5\times$	$0.8418_{\pm 0.1069}$	$0.8810_{\pm 0.0994}$	$0.1980_{\pm 0.1112}$	11.815
GO-BS	Prior LR $= 2\times$	$0.8385_{\pm 0.1085}$	$0.8850_{\pm 0.0983}$	$0.1856_{\pm 0.0943}$	11.682
GO-BS	No EMA	$0.8418_{\pm 0.1069}$	$0.8855_{\pm 0.0945}$	$0.1948_{\pm 0.1192}$	11.448
GO-BS	$n_{\text{syn}} = 100$	$0.8390_{\pm 0.1050}$	$0.8830_{\pm 0.0978}$	$0.1632_{\pm 0.0642}$	11.963
GO-BS	$n_{\text{syn}} = 500$	$0.8354_{\pm 0.1123}$	$0.8845_{\pm 0.0956}$	$0.1904_{\pm 0.1171}$	11.775
No GO-BS	**Base**	$0.8418_{\pm 0.1069}$	$0.8862_{\pm 0.0976}$	$0.1732_{\pm 0.1075}$	5.129
No GO-BS	Matched diffusion prior	$0.8326_{\pm 0.1095}$	$0.8910_{\pm 0.0947}$	$0.2784_{\pm 0.1330}$	3.558
No GO-BS	Prior epochs $= 150$	$0.8321_{\pm 0.1179}$	$0.8822_{\pm 0.0980}$	$0.2344_{\pm 0.1227}$	2.821
No GO-BS	Prior epochs $= 800$	$0.8392_{\pm 0.1208}$	$0.8842_{\pm 0.0978}$	$0.2104_{\pm 0.1022}$	13.416
No GO-BS	Prior epochs $= 1500$	$0.8418_{\pm 0.1069}$	$0.8822_{\pm 0.0986}$	$0.1564_{\pm 0.0969}$	25.185
No GO-BS	Prior LR $= 0.5\times$	$0.8418_{\pm 0.1069}$	$0.8842_{\pm 0.0968}$	$0.1724_{\pm 0.1072}$	5.436
No GO-BS	Prior LR $= 2\times$	$0.8418_{\pm 0.1069}$	$0.8853_{\pm 0.0968}$	$0.1744_{\pm 0.1087}$	5.335
No GO-BS	No EMA	$0.8418_{\pm 0.1069}$	$0.8862_{\pm 0.0976}$	$0.1732_{\pm 0.1075}$	6.208
No GO-BS	$n_{\text{syn}} = 100$	$0.8323_{\pm 0.1078}$	$0.8858_{\pm 0.0930}$	$0.1660_{\pm 0.0786}$	6.489
No GO-BS	$n_{\text{syn}} = 500$	$0.8323_{\pm 0.1174}$	$0.8862_{\pm 0.0971}$	$0.1836_{\pm 0.1276}$	5.836

Notes: Base denotes the tuned reference configuration. LR denotes the prior learning rate, EMA denotes exponential moving average, and n_{syn} denotes the synthetic sample budget. The GO-BS and no-GO-BS configurations are evaluated separately under the same repeated cross-validation protocol.

Table A21.2: **Maximum absolute change in mean TSTR AUC relative to the tuned baseline.** Values summarize the "one factor at a time" experiments in Table A21.1. Lower values indicate less sensitivity within the tested settings.

Component	GO-BS	No GO-BS
Prior family	0.0048	0.0048
Prior epochs	0.0030	0.0040
Prior learning rate	0.0045	0.0020
Synthetic sample budget	0.0025	0.0005
EMA	0.0000	0.0000

Notes: The summary follows the reported additional experiments' values. The no GO-BS synthetic budget change is reported as 0.0005; the displayed four decimal means in Table A21.1 imply 0.0004 because of numerical precision.

at most 0.0030 with GO-BS and 0.0040 without GO-BS, whereas the learning rate changes produce deviations of at most 0.0045 and 0.0020, respectively. Changes in the synthetic sample budget have smaller effects, and disabling EMA does not change the reported mean TSTR AUC at four decimal precision. These results suggest that the remaining prior and optimization controls are comparatively stable within the investigated ranges.

Computational sensitivity. Although downstream AUC varies only modestly, training cost is more sensitive to the prior epoch budget. With GO-BS, fitting time increases from 8.400 s at 150 prior epochs to 35.037 s at 1500 epochs, while mean TSTR AUC remains between 0.8825 and 0.8853 across the three tested epoch settings. Without GO-BS, the corresponding fitting times increase from 2.821 to 25.185 s, with mean AUC between 0.8822 and 0.8842. The matched diffusion prior reduces fitting time relative to the tuned baseline in both configurations, while changing the synthetic sample budget or disabling EMA has a smaller effect on mean TSTR AUC. Therefore, increasing prior training effort does not necessarily provide proportional improvements in downstream utility.

Importance of class conditional marginals. The original compact ablation in Appendix A12, Table A12.1, provides complementary evidence about the importance of class conditional marginal transformations. In that experiment, which uses TabPFN-2.5 (Grinsztajn et al., 2025) rather than the Logistic Regression evaluation employed here, disabling class conditional marginals reduces mean TSTR AUC from 0.886 to 0.872. This is the clearest utility degradation among the original

component removal settings and suggests that preserving class-specific feature distributions is particularly important. Class conditional marginals should therefore remain enabled in the default model rather than being treated as an optional optimization parameter. The two ablation protocols are complementary and should not be interpreted as a single jointly controlled experiment.

A21.2 Practical Configuration Guide

Recommended tuning priorities. The original ablation and the completed component sensitivity analysis motivate a practical tuning order. First, retain class conditional marginal transformations, which preserve class-specific feature distributions and were particularly important in the compact ablation. Second, select the block count M and construct an appropriate dependency-aware feature partition, since these choices control the representation structure and can substantially affect computational cost. Third, tune prior-specific settings, including the prior family, learning rate, training epochs, EMA, and synthetic sample budget. The relatively small AUC deviations in Table A21.2 suggest that extensive prior-specific tuning may offer limited additional utility once an appropriate structural configuration has been established.

Selecting the block budget and ordering variant. For previously unseen datasets, practitioners can initialize the block count using the "shape-scale" rule based on $\rho = m/n$ and $\kappa = nm$ in Appendix A19. The selected value may subsequently be refined using block-coherence diagnostics and held-out utility, with preference given to the smallest satisfactory budget when GO-BS computation is expensive. The choice between sample-clustered GO-BS and feature-clustered GO-BS-FC follows the Micro-Macro regime-aware selection rule in Appendix A12.1. These rules provide practical starting configurations rather than guarantees of optimality.

Block-structure diagnostics. A selected block organization can be examined using three complementary diagnostics: (i) low normalized cross-block leakage relative to within-block feature affinity, indicating that strongly dependent features are concentrated within recovered blocks; (ii) stable block assignments under bootstrap resampling, indicating that the discovered partition is not overly sensitive to individual training samples; and (iii) limited residual cross-block dependence after conditioning on the inferred block latents, indicating that the latent representation adequately explains observed dependency structure. Additional discussion of sparse latent conditional dependence, assumption violations, and recovery diagnostics is provided in Appendix A2.1. These diagnostics are recommended validation procedures and are not claimed to constitute formal tests of block identifiability.

Missingness and failure modes. For complete datasets such as Colon, the missingness mechanism requires no meaningful dataset-specific tuning. For datasets containing missing values, practitioners can compare real and synthetic per-feature missingness rates as well as class-conditional missingness patterns when labels are available. If the recovered blocks show weak coherence or poor bootstrap stability, and GO-BS does not improve held-out utility or fidelity, practitioners may reduce the cross-block leakage penalty, reconsider the block budget M, or disable GO-BS. Such decisions should be guided by validation evidence and computational cost rather than by the assumption that learned feature ordering always improves synthetic data quality.

Summary. The combined experiments support prioritizing class-conditional marginal preservation and structural configuration before extensive prior-specific optimization. Once the block budget and ordering variant are appropriately selected, moderate changes to prior-family and training controls produce relatively small variations in TSTR AUC on Colon dataset, although computational cost and raw C2ST behavior can still vary. This provides a practical configuration strategy for deploying BSTabDiff while recognizing that optimal settings and sensitivity patterns may differ across datasets and dimensionality regimes.

A22 Additional Methodological Notes and Clarifications

Selection of Tabular Generative Baselines. Our baseline selection considers recent tabular generative approaches across four families: GAN/VAE-based, diffusion/score-based, LLM-based, and HDLSS/selection-oriented methods. We examined their publicly available implementations, supported sample and feature ranges, benchmark regimes, and practical applicability to raw HDLSS data, selecting ten representative baselines from the applicable families, including the high-dimensional generator CtrTab (Li et al., 2026). Appendix A1 provides a detailed survey of these methods, while

Fig. A1.1 and Table A1.1 compare their evaluation breadth and dimensionality coverage. Among the surveyed methods, BSTabDiff is evaluated against the largest collection of baselines and covers extreme HDLSS settings involving very small sample sizes and up to $42{,}728$ raw features (Appendix A8). Although LLM-based generators are relevant to general tabular synthesis, directly representing thousands or tens of thousands of feature columns as tokens introduces substantial sequence length and attention computation costs for conventional transformer-based generators. We therefore prioritize methods applicable to direct high-dimensional tabular synthesis rather than relying on LLM-based generation at these dimensionalities. The resulting benchmark selection emphasizes methodological diversity, reproducibility, and suitability for the target HDLSS regime.

Clarity of the End-to-End Generative Pipeline. BSTabDiff integrates dependency-aware feature ordering, block-latent inference, emission estimation, and a compact generative prior into a structured synthesis pipeline. As illustrated in Fig. 1, GO-BS or GO-BS-FC estimates the feature permutation and contiguous block boundaries from training data before generator fitting. Observed continuous features are rank-Gaussianized using pooled training values, and block latents $h \in \mathbb{R}^M$ are inferred deterministically through standardized within-block averages rather than a learned encoder. Continuous emissions are fitted by least squares on observed Gaussianized entries, categorical emissions by cross-entropy, and missingness probabilities from empirical training masks, optionally conditioned on class labels. The prior $p_\theta(h \mid Y)$ then learns the joint distribution of inferred block latents through denoising score matching for diffusion or conditional maximum likelihood for flows. During generation, sampled block latents are decoded using copula-Gaussian emissions, learned inverse marginal CDFs, and categorical logits, with missingness sampled from estimated observation probabilities. Finally, the fixed learned permutation restores generated features to the original column order. Algorithms A10.1 and A10.2 in Appendix A10 detail generation and training, while Appendix A2 describes feature ordering and block segmentation.

Significance of Block-Structured HDLSS Generation. HDLSS tabular synthesis is challenging because the number of observed features can greatly exceed the available sample count, making direct estimation of high-dimensional dependence statistically difficult. The block-subunit formulation addresses this challenge by concentrating global dependence learning in a compact latent space $\mathbb{R}^M$ while retaining feature-specific probabilistic decoding in the original $\mathbb{R}^m$ space. This approach is motivated by the presence of correlated feature groups and dependency modules in scientific datasets, particularly omics and gene-expression data, and is consistent with classical sparse factor (Carvalho et al., 2008), graphical model (Meinshausen and Bühlmann, 2006), latent tree (Choi et al., 2011), and biclustering (Kluger et al., 2003) approaches discussed in Appendix A1. BSTabDiff extends this structural perspective through data-driven feature-block discovery, diffusion or flow priors, copula-based emissions, heterogeneous marginals, and explicit missingness modeling. The primary experimental evaluation in Section 4, the extension to diverse data regimes in Appendix A18, and the matched dense-factor comparison in Appendix A20 provide complementary evidence of the practical relevance and limitations of this approach.

Originality Relative to Conventional Latent Diffusion. BSTabDiff differs from a conventional latent diffusion model with an unconstrained neural encoder and decoder because its latent representation has an explicit feature-block interpretation. Rather than learning arbitrary compressed coordinates, the framework associates each latent variable h_t with a discovered block $\mathcal{S}_t$ of observed features, as specified in Definition 3.1. Each original feature is then generated through a block-conditioned probabilistic emission with a feature-specific loading, label-dependent location term, potentially heteroscedastic noise, marginal transformation, and missingness mechanism. The diffusion component is not the defining contribution but one possible prior for learning $p_\theta(h \mid Y)$; the same block-subunit formulation also supports flow, mixture, or graphical priors. Consequently, the central structural contribution is the combination of dependency-aware block discovery and constrained feature-wise probabilistic decoding, rather than applying diffusion to a generic low-dimensional embedding. The corresponding identifiability and informal sample complexity motivations are discussed in Propositions 3.3 and 3.4, subject to their stated assumptions.

Practical Construction of Feature Blocks. The feature-block organization is estimated from the training data through GO-BS, as detailed in Appendix A2. Each feature is represented as a vertex in a weighted affinity graph $G = (V, E, W)$, where edge weights quantify feature dependence using measures such as absolute correlation or other suitable affinity scores. For scalability, the graph retains the top-k strongest neighbors per feature and is symmetrized to obtain a sparse dependency representation. GO-BS then estimates a permutation $\hat{\pi}$ that brings strongly connected features closer

together, as illustrated in Fig. A2.1, and identifies $M-1$ boundaries defining contiguous feature blocks. Its objective rewards within-block affinity, penalizes cross-block leakage, and regularizes block sizes; the segmentation mechanism is illustrated in Fig. A2.2. The resulting ordering and boundaries define the canonical partition used during latent inference and generation, with synthetic observations subsequently restored to the original feature order. Although exact optimization of the GO-BS objective is NP-Hard (Theorem A2.4), the practical implementation employs a sparse approximate solver described in Algorithm A10.3. The feature-clustered alternative, GO-BS-FC, is detailed in Algorithm A10.4, with practical variant selection guidance in Appendix A12.1.

Inference of Block Latent Variables. Once the canonical feature blocks $\{\mathcal{S}_t\}_{t=1}^{M}$ are established, each training sample is represented by an M-dimensional latent vector $h=(h_1,\ldots,h_M)$. Observed continuous features are rank-Gaussianized, and each block factor h_t is estimated by averaging its observed Gaussianized coordinates, followed by standardization across samples. This lightweight factor score estimator admits a closed-form maximum-likelihood interpretation under a simplified Gaussian block model with unit feature loadings and common residual variance, where each factor minimizes the sum of squared deviations from its observed block coordinates. The inferred factors are subsequently used to fit the more general block-conditioned emissions, using least squares for continuous features and cross-entropy for categorical features on observed entries, consistent with Eq. 7. The joint prior $p_\theta(h \mid Y)$ is then trained using denoising score matching for diffusion or conditional maximum likelihood for flows. No separate inference encoder or iterative per-sample optimization is required. Algorithm A10.2 in Appendix A10 summarizes the procedure.

Learning Feature-Specific Marginal Transformations. BSTabDiff separates feature-wise marginal estimation from block-dependent latent inference. For each continuous feature j, empirical marginal CDFs $\hat{F}_{j,y}$ are estimated nonparametrically from observed training values, optionally conditioned on class $Y=y$, excluding missing entries. For latent inference, observed values are rank-Gaussianized using pooled training samples across classes: $z_{ij}=\Phi^{-1}((r_{ij}-0.5)/N_j)$, where r_{ij} is the observed rank and N_j is the number of observed values for feature j. The resulting Gaussianized coordinates are used to infer block latents and fit continuous emissions on observed entries, separating marginal behavior from block-level dependence. For generation, class-conditional inverse marginal functions $\hat{F}_{j,y}^{-1}$ are retained through sorted training observations and interpolation. Each sampled copula coordinate Z_j is mapped to $U_j=\Phi(Z_j)$ and decoded as $\tilde{X}_j=\hat{F}_{j,y}^{-1}(U_j)$, following Eqs. 3 and 4. This preserves non-Gaussian characteristics, including skewness and heavy tails, without requiring a high-capacity marginal network. The training and inverse-decoding procedures are detailed in Algorithms A10.2 and A10.1, while the contribution of class-conditional marginals is evaluated in Table A12.1 and Appendix A21.

Distinct Methodological Contributions. The defining contribution of BSTabDiff is the block-subunit generative factorization rather than the particular choice of a diffusion prior. GO-BS introduces a data-driven dependency-aware feature organization, and the resulting partition associates each block latent with a restricted subset of observed features rather than an unconstrained dense decoder. The emission model uses feature-specific scalar loadings, heteroscedastic noise, label-conditional marginal transformations, and explicit missingness mechanisms to reconstruct the full observed feature dimension from a compact latent representation. Global interactions are modeled through the joint prior $p_\theta(h \mid Y)$, so the block formulation does not require independent block latents. This architecture combines interpretable feature-block organization with a parameter-efficient emission structure whose loading complexity is $O(m)$ rather than the $O(mM)$ required by a dense latent-factor decoder. Appendix A20 provides a matched empirical comparison of these representations, while Appendix A2.1 discusses the structural assumptions, diagnostics, and possible failure modes. The resulting methodological contribution is therefore a structured generative family for HDLSS data, with diffusion serving as one interchangeable implementation of the block-latent prior.

Handling Mixed-Type Data and Missingness. Although original HDLSS benchmarks primarily contain numerical measurements, extended evaluation includes mixed-type datasets with categorical features and heterogeneous missingness patterns. Specifically, Springleaf is an HDHSS marketing dataset containing $29{,}046$ samples and $1{,}876$ features, including 22 categorical variables, high-cardinality attributes, timestamps, and substantial missingness. TCGA-BRCA contains 705 samples and $1{,}940$ features, including three categorical receptor-status variables and explicitly represented missing values. These datasets extend the evaluation to continuous and categorical features in both HDLSS and HDHSS settings, providing evidence of applicability beyond predominantly numerical

omics benchmarks. Corresponding downstream TSTR AUC results are reported in Table 4, while complementary accuracy results and broader cross-regime analyses appear in Appendix A18, including Table A18.1. Generative model's explicit missingness mechanism is defined in Eq. 2, while feature-specific decoding formulation is summarized in Definition 3.1. Together, these experiments and modeling components illustrate BSTabDiff's applicability to heterogeneous tabular settings, although the main methodological motivation remains high-dimensional generation under limited sample availability.

Computational Complexity and Practical Configuration. BSTabDiff supports both dataset-specific hyperparameter optimization and a fixed default configuration, so extensive tuning is not required for the model to operate. Appendix A16 documents the default and tuned hyperparameters, including the dataset-specific block budgets in Table A16.1 and the results of the 2000 trial Optuna (Akiba et al., 2019) search in Table A16.2. The main dataset-dependent structural choices are the number of blocks M and the optional feature ordering variant. We provide a practical initialization of M based on the feature-to-sample ratio $\rho = m/n$ and data matrix scale $\kappa = nm$ in Appendix A19, together with a completed sensitivity analysis demonstrating the trade-off between block resolution and GO-BS computational cost. A complementary regime-aware decision rule for GO-BS versus GO-BS-FC is provided in Appendix A12.1. Class-conditional marginals are treated as a core modeling component rather than an optional tuning parameter, and the missingness mechanism requires no meaningful adjustment for fully observed datasets while being estimated from observed missingness patterns otherwise. Appendix A21.2 consolidates the recommended tuning priorities, structural diagnostics, default block-budget selection, and ordering-variant choice. The Optuna (Akiba et al., 2019) experiments provide optimized baseline comparisons, whereas the default configuration offers a practical starting point for deployment without extensive dataset-specific search.

A23 Dataset and Baseline Sources

This appendix documents the datasets and baseline implementations used throughout our experiments to facilitate reproducibility and transparent evaluation. We consider 18 publicly available datasets, comprising eight initial HDLSS benchmarks, the extreme dimensional Cell Cycle RNA-seq dataset, and nine additional datasets evaluated during additional experiments, spanning HDLSS, HDHSS, LDHSS, and mixed dimensionality regimes. The collection covers omics, gene expression, face images, ResNet-50 image embeddings, high-dimensional text representations, particle physics, marketing, and census data, with sample sizes ranging from $n = 62$ to $130{,}064$ and feature dimensions ranging from $m = 14$ to $42{,}728$. Table A23.1 provides the original dataset sources and associated references, while Table A23.3 summarizes the sample and feature dimensions of all 18 datasets, including the processed Springleaf subset and CIFAR-10 image embeddings. For experimental reproducibility, Table A23.2 additionally lists the public implementations and documentation for all ten baseline methods, covering oversampling, GAN/VAE-based generation, and diffusion-based tabular synthesis.

Table A23.1: **Dataset sources and references.** Original data sources and associated references for all 18 datasets used in our experimental evaluation, including the eight original HDLSS benchmarks, Cell Cycle, and nine additional datasets evaluated during additional experiments.

Dataset	Source / Reference	Dataset	Source / Reference
Cell Cycle	Mahdessian et al. (2021); NCBI (2021)	**ORLraws10P**	Li et al. (2018, 2025)
TCGA-BRCA	Ciriello et al. (2015); Demharter and Stentoft (2020)	**CIFAR-10 (ResNet-50)**	Cukierski (2013)
Springleaf (Subset)	gonza95c et al. (2015)	**MiniBooNE**	Roe (2005)
Adult	Becker and Kohavi (1996)	**COL**	Li et al. (2018, 2025)
RELATHE	Li et al. (2018, 2025)	**GLI**	Li et al. (2018, 2025)
PCMAC	Li et al. (2018, 2025)	**SMK**	Li et al. (2018, 2025)
Lung	Li et al. (2018, 2025)	**ALLAML**	Li et al. (2018, 2025)
PRS	Li et al. (2018, 2025)	**ARC**	Li et al. (2018, 2025)
BASEHOCK	Li et al. (2018, 2025)	**TOX**	Li et al. (2018, 2025)

Table A23.2: **Baseline implementation sources.** Official implementations and documentation for the ten baseline methods used in our experimental comparisons. All repository and documentation links are clickable.

Baseline	Implementation / Documentation	Baseline	Implementation / Documentation
SMOTE	`imbalanced-learn / SMOTE`	**CTGAN**	`sdv-dev/CTGAN`
CTAB-GAN	`Team-TUD/CTAB-GAN`	**TVAE**	`sdv-dev/CTGAN`
CTAB-GAN+	`CTAB-GAN-Plus`	**TabDDPM**	`yandex-research/tab-ddpm`
ForestDiffusion	`ForestDiffusion`	**TabDiff**	`minkaixu/tabdiff`
Binary Diffusion	`binary-diffusion-tabular`	**CtrTab**	`zuqingli0404/CtrTab`

Notes: CTGAN and TVAE are two distinct baseline models provided through the same repository. SMOTE is an oversampling baseline implemented through the imbalanced-learn library. The remaining methods are linked to their respective public GitHub implementations.

Table A23.3: **Summary of all 18 datasets used in our experiments.** Here, n denotes the number of samples and m denotes the number of features. The evaluation includes eight initial HDLSS benchmarks, the extreme dimensional Cell Cycle dataset, and nine additional datasets spanning diverse dimensionality regimes.

Initial HDLSS + Cell Cycle				Additional Datasets			
Abbr	**Name**	n	m	**Abbr**	**Name**	n	m
COL	Colon	62	2,000	BRCA	TCGA-BRCA	705	1,940
GLI	GLI-85	85	22,283	CIFAR	CIFAR-10	11,000	2,048
LNG	Lung	203	3,312	MB	MiniBooNE	130,064	50
PRS	Prostate	102	5,966	SPR	Springleaf (Subset)	29,046	1,876
SMK	SMK	187	19,993	ORL	ORLraws10P	100	10,304
TOX	TOX171	171	5,748	ADULT	Adult	48,842	14
AML	ALLAML	72	7,129	BASE	BASEHOCK	1,993	4,862
ARC	Arcene	200	10,000	REL	RELATHE	1,427	4,322
CC	Cell Cycle	1,067	42,728	PCMAC	PCMAC	1,943	3,289

Notes: The first eight datasets are the original HDLSS benchmarks. Cell Cycle is an RNA-seq dataset with the highest feature dimensionality ($m = 42{,}728$) in our evaluation. CIFAR-10 uses 2,048-dimensional ResNet-50 image embeddings rather than raw pixels. Springleaf refers to the processed subset. The nine additional datasets include omics, image embeddings, particle physics, marketing, census, face image, and text classification data.

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 introduction accurately summarize BSTabDiff's main contributions, including the HDLSS ($n \ll m$) block-subunit generator, compact block-latent priors, Graph-guided Ordering with Block Segmentation (GO-BS), theoretical analysis, empirical evaluation, and energy-carbon analysis. The primary claims focus on HDLSS tabular generation, while additional experiments examine broader dataset regimes and practical limitations. We do not claim universal superiority across all data regimes.

   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: The paper includes a dedicated limitations discussion before the conclusion. It acknowledges that BSTabDiff is primarily designed for High-Dimensional Low-Sample Size (HDLSS; $n \ll m$) tabular data, while additional experiments explore other dataset regimes (e.g. HDHSS, LDHSS, MixedRegime). It also discusses the dependence of Graph-guided Ordering with Block Segmentation (GO-BS) on meaningful local feature dependence structure, the potential computational overhead of feature ordering, and the possibility of reduced benefits when such structure is weak. The paper does not claim universal superiority across all dataset regimes.

   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: [Yes]

   Justification: The paper states the relevant assumptions and provides proofs or proof sketches for its theoretical results, including identifiability, within-block SNR scaling, GO-BS NP-hardness, exact recovery under block dominance, and sparse computational complexity. The formal GO-BS results are detailed in Appendix A2. The identifiability result is accompanied by a proof sketch, while the sample complexity advantage is explicitly labeled as informal and intended to provide theoretical intuition rather than a complete formal guarantee.

   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 paper provides dataset descriptions and original source links in Table A23.1, along with baseline implementation sources in Table A23.2, evaluation protocols, cross-validation setup, hyperparameter tuning procedures, compute environment, and implementation details needed to reproduce the main experimental results. The supplemental material includes the BSTabDiff implementation and most experimental `.ipynb` notebooks with executed outputs and reported results. The notebooks have been cleaned and validated through reproducibility smoke tests, including installation and execution using the released PyPI package. Additional hyperparameters, pseudocode, energy-measurement protocols, and runtime/resource details are provided in the appendix. The implementation and reproducibility resources are publicly available on GitHub (`https://github.com/zadid6pretam/BSTabDiff`) and PyPI (`https://pypi.org/project/bstabdiff`), with installation supported via `pip install bstabdiff`.

   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: The datasets used in this study are publicly available through their original sources, with access links provided in Table A23.1. Baseline implementations and their source links are documented in Table A23.2. We provide a supplemental code package containing the BSTabDiff implementation and most experimental `.ipynb` notebooks with executed outputs and reported results. The notebooks have been cleaned and validated through reproducibility smoke tests, including installation and execution using the released PyPI package. Detailed experimental settings, hyperparameters, pseudocode, and evaluation protocols are provided in the appendix. The source code and reproducibility resources are publicly available on GitHub (`https://github.com/zadid6pretam/BSTabDiff`) and PyPI (`https://pypi.org/project/bstabdiff/`), with installation supported via `pip install bstabdiff`. The datasets are accessed through their original sources rather than redistributed.

   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: The paper specifies the training and evaluation procedures, including publicly available datasets, baseline implementations, the 5×5 repeated cross-validation protocol, training and test splits, TSTR/TRTR evaluation, and Optuna-based hyperparameter optimization with 2000 trials. Detailed hyperparameter configurations, search spaces, optimization procedures, pseudocode, compute environment, energy-measurement protocols, and runtime/resource information are provided in the appendix. The supplemental code package and experimental notebooks further document the implementation and experimental settings, supporting reproducibility of the reported results.

   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: The main TSTR/TRTR results are reported as mean $\pm$ standard deviation over strict 5×5 repeated cross-validation (25 evaluations), capturing variability across train/test splits. Additional ablation studies follow the same protocol where applicable, while fidelity diagnostics report complementary measures, including KS, W1, correlation discrepancy, mutual information (MI) discrepancy, privacy distance, and C2ST. Statistical significance is further assessed in Appendix A15 using a Friedman omnibus test, followed by pairwise one-sided Wilcoxon signed-rank tests with Holm correction, as reported in Table A15.1.

   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: The paper reports the experimental compute environment, including the Vulcan cluster equipped with NVIDIA RTX A6000 GPUs, Intel Xeon Gold CPUs, and 503 GB of system RAM. Table 1 provides per-dataset execution time, GPU memory usage, CPU Resident Set Size (RSS), energy consumption, and CO_2e emissions for the main BSTabDiff runs. Additional computational resource analyses, runtime measurements, and energy-carbon diagnostics are provided in Appendices A9, A17, A3, A5, and A7, including CodeCarbon (Courty et al., 2026)-based measurements and GO-BS/GO-BS-FC computational analyses. These details document the hardware requirements and computational costs associated with reproducing the experiments.

   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: We have reviewed the NeurIPS Code of Ethics (`https://neurips.cc/public/EthicsGuidelines`) and confirm that our research conforms to its principles. The study uses publicly available datasets with appropriate citations and source links, does not involve new human-subject data collection, deception, or intervention, and reports methodological limitations, computational requirements, and energy-carbon impacts. We also provide publicly accessible code and reproducibility resources through GitHub and PyPI, supporting transparency and responsible research practices.

   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: [Yes]

    Justification: The paper includes a dedicated Limitations and Impact Statement discussing both positive and potential negative societal impacts. Positive impacts include synthetic data generation for scarce HDLSS and sensitive scientific domains, improved data augmentation and benchmarking, and transparent reporting of energy consumption and CO_2e emissions. Potential risks include privacy leakage, over-reliance on synthetic data, propagation of existing biases, and inappropriate use in high-stakes applications. We emphasize that

synthetic data should complement rather than replace real data validation, domain expert review, privacy assessments, and appropriate safety checks before downstream deployment.

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 paper does not release high-risk pretrained language models, image generators, or scraped datasets. The experiments use publicly available, properly cited tabular datasets, with original sources documented in Table A23.1. The released materials consist of the BSTabDiff implementation, experimental notebooks, and reproducibility instructions rather than sensitive datasets or pretrained models intended for unrestricted deployment. We do not redistribute the original datasets and acknowledge potential privacy, misuse, and downstream validation risks in the Limitations and Impact Statement. Users are encouraged to assess privacy and data utility before applying BSTabDiff to sensitive or high-stakes domains.

    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: The paper appropriately cites the original datasets, baseline implementations, libraries, and tools, with source links provided in Tables A23.1 and A23.2. The supplemental README documents asset sources and licensing information, and the released BSTabDiff code includes its own license. We respect the original assets' licensing and terms of use.

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: BSTabDiff is documented through its project webpage (`https://www.zadidhabib.com/bstabdiff.html` or alternatively `https://zadid6pretam.github.io/bstabdiff`), GitHub repository (`https://github.com/zadid6pretam/BSTabDiff`), PyPI package (`https://pypi.org/project/bstabdiff/`), Hugging Face Space (`https://huggingface.co/spaces/zadid6pretam/BSTabDiff-Studio`), and supplemental code package, which provide the implementation, interactive demonstration, experimental notebooks, and reproducibility instructions. The paper and appendix further document the methodology, pseudocode, hyperparameters, computational requirements, and limitations.

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 crowdsourcing experiments, new human-subject data collection, participant studies, or compensated human labor. All experiments use publicly available, cited tabular datasets.

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 crowdsourcing, new human-subject data collection, participant studies, or interventions with study participants. All experiments use publicly available, cited tabular datasets, so IRB approval or equivalent human-subjects review is not applicable.

    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: The core method development, theory, experiments, and evaluation of BSTab-Diff do not use Large Language Models (LLMs) as an important, original, or non-standard component. LLM assistance was limited to writing, editing, LaTeX formatting, and polishing repetitive experimental notebook code, without altering the core methodology, scientific claims, or experimental results.

    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.