TABMOSAIC: UNIFIED IN-CONTEXT DIFFUSION FOR TABULAR GENERATIVE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative modeling for tabular data has advanced to the point where models can synthesize realistic records, recover missing values, and generate samples conditioned on class labels or observed attributes. Despite recent advances, existing approaches either require table-specific training or restrict in-context learning to single-cell prediction. In this work, we formulate these tabular generative tasks as a single problem: modeling the joint conditional distribution of an arbitrary set of target cells given the observed cells of an unseen table. To this end, we turn to diffusion, whose denoising objective is defined jointly over all target cells rather than one cell at a time. Under the pretraining distribution, this objective is minimized by the posterior predictive distribution over the target set. Building on this, we propose TabMosaic, a unified in-context diffusion model for generating mixed-type target cells in unseen, potentially incomplete tables without table-specific training. We further show that its pretraining objective bounds the conditional negative log-likelihood and the divergence from the posterior predictive for arbitrary target sets, extending the Bayesian account of in-context learning from a single-value prediction to joint multi-cell generation. Across 48 real-world tables, TabMosaic ranks first on all evaluated generative tasks against 20 baseline approaches.

1 INTRODUCTION

In domains such as healthcare, finance and scientific discovery, tabular data is widely used for data-driven decision-making (Borisov et al., 2022a; Van Breugel & Van Der Schaar, 2024). Recent tabular generation models synthesize realistic rows, recover missing values, and generate samples conditioned on classes or attributes (Kotelnikov et al., 2023; Kim et al., 2023; Zhang et al., 2024; Shi et al., 2025; Zhang et al., 2025a; Jarrett et al., 2022; Du et al., 2024; Zhang et al., 2025b). Yet these models are typically dataset-specific: they must be trained or fine-tuned separately for each new table, requiring a new training run whenever the dataset changes. This motivates a natural question:

Can a single pretrained model perform diverse tabular generative tasks on unseen tables without table-specific training, using only in-context information?

Recent work has been moving toward this goal by transferring tabular models across datasets. Cross-table generative models pretrain across many tables, reducing the need to learn each dataset entirely from scratch, but still require fine-tuning on each new one (Lin et al., 2025; Jiang et al., 2026). In-context tabular models remove this adaptation step and operate directly on unseen tables (Hollmann et al., 2025; Qu et al., 2025). However, these models are trained for individual prediction targets rather than the joint distribution over an arbitrary set of cells, so multi-cell generation and imputation must be assembled from separate predictions rather than modeled directly. No existing work provides a single pretrained model that learns the joint conditional distribution over arbitrary target sets and applies it directly to unseen tables without table-specific adaptation.

Our contributions. In this work, we formulate tabular generation, imputation and conditional generation as a single problem: learning the joint conditional distribution over arbitrary target sets on unseen tables. This formulation shifts tabular generative modeling from training a new model for each dataset and task to learning a reusable conditional model once across datasets. Different generative tasks are specified simply by which cells are observed and which are queried.

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

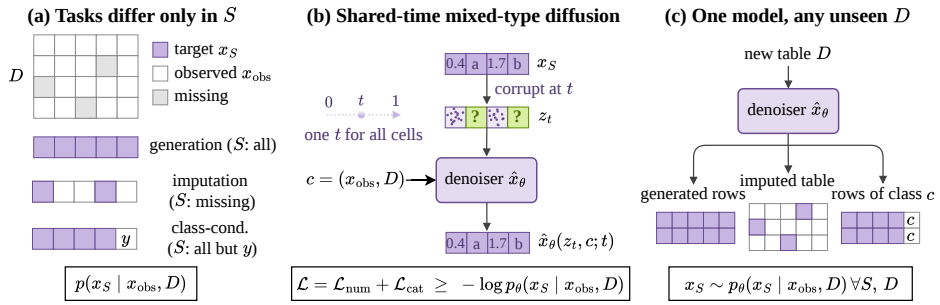

Figure 1: **Overview of our proposed framework.** (a) A unified formulation expresses generation, imputation and class-conditional generation as the same conditional query, only differing in the target set S. (b) A shared-time diffusion objective jointly models mixed numerical and categorical targets. (c) Our TabMosaic applies directly to unseen tables without table-specific training.

To this end, we *first* formulate a unified learning objective for tabular generative tasks. Generation, imputation, and conditional generation are treated as the same family of conditionals, differing only in which cells are designated as targets. We instantiate this objective with in-context diffusion over arbitrary sets of mixed numerical and categorical target cells, without imposing an autoregressive order. We show that, under the pretraining prior, its population optimum is the posterior predictive distribution. We derive a variational upper bound on the conditional negative log-likelihood. Under the pretraining prior, the bound also controls the KL divergence from the posterior predictive distribution to the model distribution, up to a model-independent constant.

Second, we introduce TabMosaic, an in-context diffusion model that realizes this objective on unseen tables. The model jointly captures dependencies among numerical and categorical cells while conditioning on the observed cells and context. To scale TabMosaic to long and incomplete tables, we develop a two-staged pretraining strategy: the model first learns from short, complete tables and is then extended to long, partially observed contexts. In the second stage, multiple corruptions share a single context encoding, reducing computation and gradient variance. Missing context cells are represented explicitly, so the model can condition directly on incomplete tables at inference time.

Third, we evaluate TabMosaic on generation, imputation, and class-conditional generation across the 48 datasets from a recent tabular benchmark (Erickson et al., 2026). Against 20 baselines spanning dataset-specific generators and imputers, in-context predictors, and training-free conditional samplers, our single pretrained TabMosaic *ranks first* across all three tasks. We also compare TabMosaic against the exact posterior predictive distribution on synthetic priors for which the posterior is available in closed form. For imputation, we evaluate performance under different missingness patterns and mechanisms, such as settings where entire columns are missing. We conduct an ablation study to assess the contribution of the staged pretraining strategy.

2 Background and Related Work

2.1 In-context learning as posterior prediction

A sequence model trained by maximum likelihood on data drawn from a mixture over latent tasks approximates, at its optimum, the Bayesian posterior predictive under that mixture: conditioning on a prompt plays the role of conditioning on observations, and the model marginalizes the latent task rather than inferring it explicitly. This is the reading of in-context learning as implicit Bayesian inference (Xie et al., 2021), and the basis of a line of work that trains sequence models on synthetic priors and compares them with the Bayes-optimal predictor on tasks whose posterior is available in closed form (Garg et al., 2022; Raventós et al., 2023; Panwar et al., 2024). Tables fit this view, since the rows of a table are exchangeable draws from an unknown mechanism, and prior-data fitted networks (Müller et al., 2022) make it concrete for one cell: a transformer pretrained on synthetic tables drawn from a prior $p(\phi)$ over data-generating mechanisms approximates, for a query row with features x, label y, and context rows D, the posterior predictive distribution under the prior:

$$p(y \mid x, D) \;=\; \int p(y \mid x, \phi)\; p(\phi \mid D)\; d\phi \tag{1}$$

The same argument applies to any target set x_S, provided that the pretraining objective is the likelihood of x_S itself.

2.2 Generative models for tables

Per-dataset models. Diffusion models pair a fixed corruption process with a learned reverse process, Gaussian for continuous data (Ho et al., 2020; Song et al., 2021; Karras et al., 2022) and masked for discrete data (Austin et al., 2021; Sahoo et al., 2024); GIDD (von Rütte et al., 2025) interpolates the mask with uniform noise, which yields a self-correcting sampler with an exact ELBO, and we use the EDM (Karras et al., 2022) and GIDD kernels (§3.2). For tables, generators of this kind are trained per dataset, in score-based, latent, mixed-type continuous-time diffusion model forms (Kim et al., 2023; Zhang et al., 2024; Shi et al., 2025) or with gradient-boosted trees in place of a network (Jolicoeur-Martineau et al., 2024); TabNAT pairs a masked transformer for discrete columns with a diffusion loss for continuous ones (Zhang et al., 2025a). Per-dataset imputers follow the same pattern (Van Buuren & Oudshoorn, 1999; Stekhoven & Bühlmann, 2012; Yoon et al., 2018; Jarrett et al., 2022; Du et al., 2024; Zhang et al., 2025b). Each recovers the conditionals over a target set in one of two ways: i) Unconditional models are conditioned after training by inpainting, which fixes the given cells at every reverse step; the result approximates the conditional of the trained joint rather than sampling it, and the released procedures cover a single target column or cells of one type (Zhang et al., 2024; Shi et al., 2025; Jolicoeur-Martineau et al., 2024); TabDDPM instead conditions on one column designated before training, the class label (Kotelnikov et al., 2023). ii) TabNAT (Zhang et al., 2025a) and GReaT Borisov et al. (2022b) train conditionals of a random target set given the rest by autoregression over a random order, which imposes an order on unordered set of columns; each order is a different factorization, and no term of the objective makes them agree. Both are trained on a complete table, so the context itself cannot contain missing cells.

Pretraining across tables. CTSyn (Lin et al., 2025) pretrains an autoencoder into a shared latent space and a schema-conditioned latent diffusion model across tables, and fine-tunes both on each new table; TabFORGE (Jiang et al., 2026) likewise fine-tunes per table. Both target unconditional generation, and neither answers a query on an unseen table without a gradient update.

In-context predictors. Models pretrained to predict single cells also impute and generate without training on the table: missing cells are filled by chaining a point predictor over them, and rows are generated by chaining the predictor over the columns, optionally followed by rounds of masking and refilling (Hollmann et al., 2025; Zhang et al., 2025c). The objective is a sum of per-cell log-likelihoods given the unmasked ones, so no term involves a target set as a whole, and a generated row is a chain in a chosen order.

Energy samplers on a classifier. TabEBM (Margeloiu et al., 2024) and TabPFGen (Ma et al., 2024) use the logits of a pretrained tabular classifier as an energy and draw class-conditional rows by Langevin dynamics; TabPFGen also imputes by running the same Langevin dynamics on the missing cells only, with the observes cells held fixed and the label given. The classification objective constrains only the class posterior and leaves the distribution over the cells free (Grathwohl et al., 2019), every query must supply a label, and the model has no likelihood.

3 A Unified Objective for Tabular Generative Tasks

Setting. A tabular task instance consists of a *context* set $D = \{r_1, \ldots, r_n\}$ of n rows and a query row $x = (x_1, \ldots, x_F)$ over the same F columns, where each $r_i \in \mathbb{R}^F$ is a row of the table whose cells may be missing, and the cells of x split into an observed part x_{obs} and a *target set* $S \subseteq \{1, \cdots, F\}$ of cell indices of the row, $S = S_{\text{num}} \cup S_{\text{cat}}$ of numerical and categorical indices. Multiple query rows are handled identically and independently given D, and therefore we describe a single row. We write $c = (x_{\text{obs}}, D)$ for the conditioning information. Note that the tasks we consider are instances of a *single* distributional object, the conditional joint distribution over the target cells, and they differ only in the choice of S, as shown in Fig. 1 (a).

3.1 One family of tasks: conditional queries over cell sets

We pose all of the tabular generative tasks above as learning the family $\{\, p(x_S \mid x_{\text{obs}}, D) \,\}_S$ with one model. Under a prior $p(\phi)$ over data-generating mechanisms, where the rows of D and the

query row are drawn i.i.d. from $p(\cdot \mid \phi)$, the ideal member of this family is the posterior predictive distribution of Eq. 1, generalized from a single value to an arbitrary cell set:

$$p(x_S \mid x_{\text{obs}}, D) \;=\; \int p(x_S \mid x_{\text{obs}}, \phi)\; p(\phi \mid D)\; d\phi. \tag{2}$$

Learning this family imposes three requirements, which drive our design: i) a valid *joint* distribution for every mixed-type target set S; ii) a *single* pretrained network that serve every (S, x_{obs}) in-context; and iii) a training objective that provably targets Eq. 2.

3.2 In-context joint diffusion over cell sets

Forward process. The diffusion state at time $t \in [0,1]$ is $z_t = (z_t^{\text{num}}, z_t^{\text{cat}}) \in \mathbb{R}^{|S_{\text{num}}|} \times \prod_{j \in S_{\text{cat}}} \mathcal{V}_j$, where $\mathcal{V}_j$ is the class set of column j augmented with a MASK token. We corrupt only the target cells; the observed cells and the context remain clean. Given the clean values x_S, the forward marginal factorizes over cells on a *shared* time axis, i.e., a Gaussian (EDM) kernel on numerical cells and a GIDD kernel on categorical cells:

$$q_t(z_t \mid x_S) = \prod_{j \in S_{\text{num}}} \mathcal{N}\left(z_{t,j};\, x_j,\, \sigma(t)^2\right) \;\cdot\; \prod_{j \in S_{\text{cat}}} \text{Cat}\left(z_{t,j};\, \alpha_t e_{x_j} + \beta_t \pi_t\right), \tag{3}$$

where $\sigma(t)$ is the EDM noise schedule, $e_{x_j} \in \mathbb{R}^{|\mathcal{V}_j|}$ is the one-hot vector of the clean class x_j, which is never MASK, $\beta_t = 1 - \alpha_t$, and π_t is the mixing distribution of GIDD (von Rütte et al., 2025), a mixture of MASK and uniform noise whose uniform component lets the reverse process revise cells that have already been unmasked (See Appendix B).

Reverse process. A single denoiser $\hat{x}_\theta(z_t, c; t)$ reads the *entire* mixed state together with the conditioning $c = (x_{\text{obs}}, D)$, and outputs a scalar estimate for each numerical target cell and a class distribution for each categorical target cell. On a time grid $1 = t_N > \cdots > t_0 = 0$, each reverse step is the posterior of the forward process with the clean target value replaced by the denoiser output, and every factor is conditioned on the full mixed state:

$$p_\theta(z_s \mid z_t, c) = \prod_{j \in S} q\big(z_{s,j} \mid z_{t,j},\, \hat{x}_{\theta,j}(z_t, c; t)\big). \tag{4}$$

The model is the composition of the reverse steps, started from the terminal distribution $p(z_{t_N})$, which is independent of the data (Gaussian noise on numerical cells, MASK on categorical cells):

$$p_\theta(x_S \mid c) = \int p(z_{t_N}) \prod_{i=N}^{1} p_\theta\big(z_{t_{i-1}} \mid z_{t_i}, c\big)\, dz_{t_1:t_N}. \tag{5}$$

Proposition 1 (Validity)**.** *For every S, c, and θ, Eq. 5 defines a normalized probability distribution on the mixed product space. The proof is in Appendix A.*

We highlight three properties of the proposed construction. i) The joint is factorized along the *time axis*, not along a cell or modality ordering, so no ordering is chosen and the two modalities are coherent by construction (Proposition 1). ii) Dependencies between cells, including across modalities, arise from conditioning: each factor of Eq. 4 reads the current state of every other cell, and the composition over steps transports this information. iii) With a single step (N=1) the model reduces to the conditionally independent product $\prod_{j \in S} p_\theta(x_j \mid c)$; the iteration is what creates the coupling.

3.3 Training objective and guarantee

Target patterns. During pretraining the target pattern is drawn independently of the cell values and of t: with probability 0.4 all query cells are targets; otherwise each cell is a target independently with rate $\rho \sim \mathcal{U}(0,1)$. A single t then corrupts all target cells. Drawing S and t independently is valid because the bound of Theorem 1 holds for every target set separately, i.e., S only selects which cells enter it and t ranges over $[0,1]$ for each S. As a result, every combination of target set and noise level lies inside the training distribution, so a query can start its reverse process at any t and for any fraction of target cells.

Loss. Let $\lambda(\sigma)$ be the EDM weight and $w(z_{t,j})$ be the state-dependent GIDD weights. Our pre-training loss is defined as follows:

$$\mathcal{L} = \mathbb{E}_{(x\cup D)\sim p(\mathcal{D})}\, \mathbb{E}_S\, \mathbb{E}_t\, \mathbb{E}_{z_t\sim q_t}\big[\, \tfrac{1}{|S|}\,(\mathcal{L}_{\text{num}} + \mathcal{L}_{\text{cat}})\,\big], \tag{6}$$

$$\mathcal{L}_{\text{num}} = \sum_{j\in S_{\text{num}}} \lambda(\sigma(t))\big(\hat{x}_{\theta,j} - x_j\big)^2, \qquad \mathcal{L}_{\text{cat}} = \sum_{j\in S_{\text{cat}}} w(z_{t,j})\big[D_{\text{KL},j} + D_{\text{IS},j}\big], \tag{7}$$

where $D_{\text{KL},j}$ and $D_{\text{IS},j}$ are the divergence terms of the GIDD kernel. We emphasize that the per-cell sums are *not* a factorization of the model: they arise since the per-step distributions in Eq. 3 and Eq. 4 factorize given z_t, whereas the joint distribution in Eq. 5 does not.

Theorem 1 (Conditional ELBO)**.** *For fixed (x, S, c), let $\ell^\star_\theta(x, S, c)$ be the inner expectation $\mathbb{E}_t\, \mathbb{E}_{z_t}[\cdot]$ of Eq. 6 without the normalization by $|S|$ and with the numerical weight λ replaced by the likelihood weight $w_{\text{lik}}(t) = \sigma'(t)/\sigma(t)^3$; the categorical weights are already those of the likelihood bound. Then $\ell^\star_\theta(x, S, c) \geq -\log p_\theta(x_S \mid c) - C$, where C is independent of θ, i.e., $-\ell^\star_\theta - C$ is an evidence lower bound on $\log p_\theta(x_S \mid c)$. The proof is in Appendix A.*

Corollary 1 (The objective targets the posterior predictive distribution)**.** *Let tables be drawn from the prior and S independently of the cell values. Then*

$$\mathbb{E}\big[\mathrm{KL}\big(p(x_S \mid x_{\text{obs}}, D)\,\big\|\, p_\theta(x_S \mid x_{\text{obs}}, D)\big)\big] \;\leq\; \mathbb{E}\big[\ell^\star_\theta\big] - C',$$

where $p(x_S \mid x_{\text{obs}}, D)$ is the posterior predictive of Eq. 2 and C' is independent of θ. Moreover, $\mathcal{L}$ and $\mathbb{E}[\ell^\star_\theta]$ are minimized by the same denoiser: $\hat{x}_{\theta,j} = \mathbb{E}[x_j \mid z_t, c]$ for numerical cells and the posterior PMF $p(x_j \mid z_t, c)$ for categorical cells. The proof is in Appendix A.

Corollary 2 (The posterior predictive minimizes the objective)**.** *Under the assumptions of Corollary 1, the denoiser of the posterior predictive of Eq. 2, i.e., $\hat{x}^\star_j = \mathbb{E}[x_j \mid z_t, c]$ for numerical and $p(x_j \mid z_t, c)$ for categorical cells, minimizes $\mathcal{L}$ and $\mathbb{E}[\ell^\star_\theta]$, and with it the model of Eq. 5 equals the posterior predictive as $N \to \infty$. The proof is in Appendix A.*

Corollary 1 bounds the gap to the posterior predictive by the objective, and Corollary 2 shows that the posterior predictive attains it, i.e., it is the minimizer and not only the target. Together they make the Bayesian reading of in-context learning exact for our objective, with the cross-entropy identity relaxed to an upper bound through the diffusion ELBO. The condition on S holds by construction, since every target pattern we draw is independent of the cell values, which makes the target the posterior predictive rather than a conditional that also depends on which cells are hidden.

4 TabMosaic: A Unified In-Context Diffusion Model

4.1 Architecture

The denoiser operates at the resolution of individual cells, so that every cell of the context and the query can be a target, an observation, or missing. Query rows attend to the context only, so they are answered independently and in parallel, as Eq. 2 is defined for one row at a time.

Tokenization. Each cell of the context and the query row is one token: a categorical cell by a lookup of its class (or MASK), a numerical cell by a linear map of its value. A learned *observation-state* embedding marks the cell as a target (a cell of x_S), observed (a cell of x_{obs} or of D), or missing in the context (a placeholder value with a missing flag), so the target set can differ across rows within one forward pass. The time embedding is added to target tokens only, so the context representation does not depend on t. Consequently, the keys and values of the context tokens are reused across denoising steps (§4.3) and the multiple corruptions (§4.2). No token carries a column index or a role such as "label": the label is an ordinary column.

Backbone. We follow the two-axis attention of in-context tabular transformers (Hollmann et al., 2025), which alternates between the cells of a row and the rows of a column. We differ in the following aspects: i) Tokens carry no column-specific embedding, neither random per-column embeddings nor positional encodings over columns, so the network is permutation-equivariant in the columns. ii) Instead of a single label head, a per-cell head, shared by all cells, returns a scalar for each numerical target and class logits for each categorical target.

4.2 PRETRAINING: PRIOR, ESTIMATOR, AND STAGES

Synthetic prior. We draw tables from a structural causal model (SCM) prior of TabICL (v2) (Qu et al., 2026) with two changes that follow from §3.1: a label is an ordinary column, whose cardinality follows the same sampling scheme as the categorical features, and half of the tables have a numerical label.

Multi-corruption estimator. The loss of Eq. 6 is a nested expectation: over tables and target sets, and, inside, over the corruption $z_t \sim q_t(\cdot \mid x_S)$. One corruption per table estimates the inner expectation without bias but with noise. To reduce this noise, for each table we draw one pair (t, S) and K independent corruptions $z_t^{(1)}, \dots, z_t^{(K)} \sim q_t(\cdot \mid x_S)$, and average their losses,

$$\hat{\ell}_K(x, S, c) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|S|} \big[\mathcal{L}_{\text{num}} + \mathcal{L}_{\text{cat}}\big]\big(z_t^{(k)}\big), \tag{8}$$

which is unbiased for the inner expectation and divides the variance due to the corruption by K. The cost is far below K times that of one corruption: since query rows attend only to the context, the K copies are packed along the row axis and share one context encoding, which is most of the cost of a step in Stage 2, so only the query-side compute grows with K. Averaging the outputs instead of the losses would change the objective (See Appendix C).

Two stages. Pretraining proceeds in two warm-started stages, each widening the training distribution and adding a capability. T is the number of rows per table, F is the number of features.

1. **Stage 1: joint on short, complete tables.** Each table has $T = 1{,}024$ rows and $F \sim \log\mathcal{U}(1, 100)$ features, and the context fraction is sampled from $\mathcal{U}(0.3, 0.9)$. Tables have no missing values. We train on 32M tables with a learning rate of 8×10^{-4}. Multi-corruption is not activated ($K = 1$).
2. **Stage 2: long and incomplete contexts.** We sample $T \sim \log\mathcal{U}(400, 10{,}240)$ rows and then $F \sim \log\mathcal{U}(4, \min(200, B/T))$ features under a cell budget $B = 250{,}000$, since the cost of the denoiser grows with $T \times F$. The context fraction is fixed to 0.8. Context cells are hidden under MCAR, MAR, or MNAR at a per-table rate $\rho \sim \mathcal{U}(0, \rho_{\max})$, where $\rho_{\max}$ increases linearly from 0 to 0.2 over the first quarter of the stage. We train on 2.56M tables with a learning rate of 10^{-4} and $K = 4$.

Neither stage changes the minimizer of Eq. 6: missing context cells enter as inputs, never as targets, so with an incomplete context D stands for its observed cells and their missingness pattern (See Appendix D). Optimizer and other constants are in Appendix E.

4.3 INFERENCE: ONE PROCEDURE FOR EVERY TASK

Because every generation task we consider is a conditional query against the same distribution, no task needs a procedure of its own: Algorithm 1 answers all of them with the same pretrained network, and a task is an instantiation of which cells form the target set S, where the reverse chain starts (t_{start}), how many steps it takes (N), and how many trajectories it draws (M). Generation observes nothing and targets the whole row; imputation observes part of the row and target the rest; class-conditional generation observes the label and targets the rest. Per-task defaults are in Appendix F.

Algorithm 1: Unified conditional-query answering

Require: $c = (x_{\text{obs}}, D)$, S, grid $t_N > \cdots > t_0 = 0$, M

for $m = 1, \dots, M$ **do**

  $z_{t_N,j}^{(m)} \leftarrow \begin{cases} \text{MASK}, & j \in S_{\text{cat}} \\ \sigma(t_N)\,\varepsilon_j, & j \in S_{\text{num}} \end{cases}$

  for $i = N, \dots, 1$ **do**

    $z_{t_{i-1}}^{(m)} \sim p_\theta(\cdot \mid z_{t_i}^{(m)}, c)$ ▷ Eq. 4

  end for

end for

return $\{z_{t_0}^{(m)}\}_{m=1}^{M}$

Two properties make every instantiation valid and cheap (See Appendix F). i) *The chain may start at any* $t_{\text{start}} \leq 1$: since S and t are independent in pretraining, every target set was seen at every noise level, and MASK lies in the support of every categorical marginal. ii) *The context is encoded once*: query rows attend to the context but not the reverse, and the t enters only the target cells, so the context keys and values are reused across steps, query rows, and trajectories.

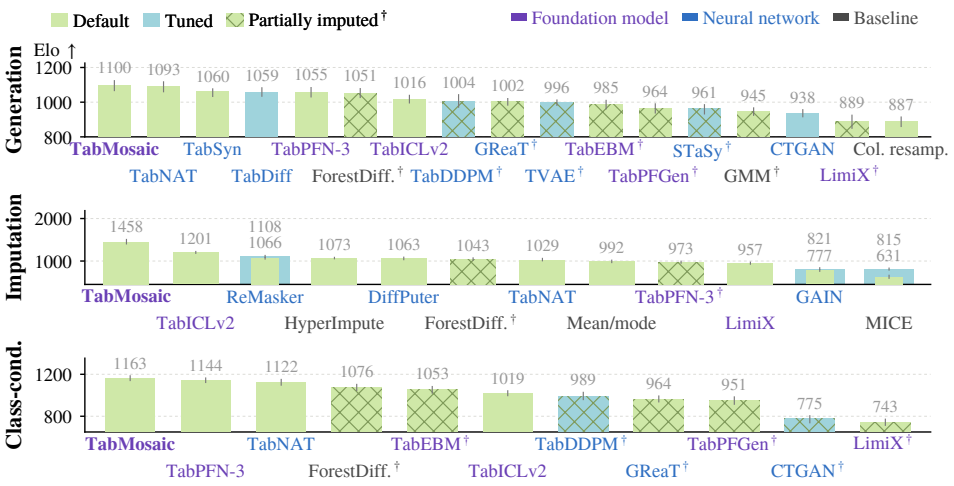

Figure 2: Elo with 95% bootstrap intervals for generation and imputation on the 48 tables and class-conditional generation on the 36 classification tables. Bar color marks default and tuned entries, name color the model family, and hatching (†) a method that did not complete every table, whose missing tables are filled with the reference entry. Full results are in Appendix L.

5 Evaluation

We evaluate TabMosaic through the following research questions: **(RQ1. Generative performance)** How does a single pretrained model without per-table training or tuning compare with models trained separately for each table on the three tasks? **(RQ2. Posterior predictive fidelity)** How closely does the learned joint distribution match the posterior predictive distribution in Eq. 2, and how does this compare with the product of exact per-cell posterior predictives? **(RQ3. Target-set performance)** How does TabMosaic compare with per-dataset and in-context baselines as the target set varies in missing rate, mechanism, and pattern? **(RQ4. Staged pretraining)** How does the second pretraining stage affect performance on long and incomplete tables?

5.1 Experimental setup

Datasets. All real-data experiments use 48 tables from TabArena (Erickson et al., 2026) with the official splits; every stochastic method runs with three seeds, and methods are ranked by the official Elo procedure (Appendix G).

Metrics. All three evaluations synthesize as many rows as the training split and score them by utility (train on synthetic, test on real relative to the training on real), per-column W_1 and TV distances, and, for generation, α-precision and β-recall (Alaa et al., 2022). *Imputation* hides 30% of the feature cells that are not already missing, completely at random, so every method fits on or conditions on the incomplete table; it adds min–max-scaled RMSE and categorical error on the hidden cells. *Class-conditional generation* gives the training split's own label column as the condition; it adds label consistency, the fraction of rows that a classifier trained on real data assigns to the requested class, and scores W_1 and TV per class.

Baselines. Per-table models are tuned over ten configurations where their paper tunes and run with their published configuration otherwise. For generation we compare with CTGAN, TVAE, TabDDPM, STaSy, TabSyn, TabDiff, TabNAT, ForestDiffusion, and GReaT; for imputation with a mean/mode imputer, MICE, GAIN, HyperImpute, ReMasker, DiffPuter, TabNAT, and ForestDiffusion; and for class-conditional generation with TabDDPM, CTGAN, GReaT, ForestDiffusion, TabNAT, and the training-free TabEBM and TabPFGen, each conditioning by its own mechanism (See Table 4). TabMosaic is one pretrained model used as is for every task and table. In every task we also enter the in-context predictors TabPFN-3, TabICLv2, and LimiX with their official generation and imputation procedures (Appendix H).

5.2 RQ1. Generative performance

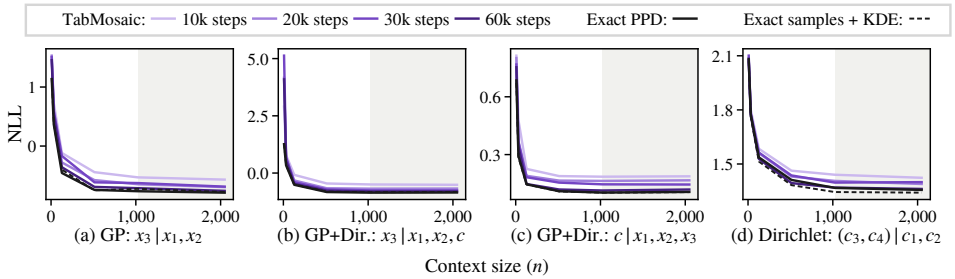

Figure 3: NLL of held-out cells against the context size n, for a small model pretrained with Eq. 6 on priors with a closed-form posterior predictive (x: numerical, c: categorical cell; Appendix I). Solid black: exact posterior predictive; dashed: M exact samples under the same KDE as the model, i.e., the floor for a sampler. Shaded: beyond the context sizes seen in pretraining.

Generation. TabMosaic ranks first by Elo (1104), with the best mean rank in utility (3.9, against 5.0 for TabNAT), i.e., the rows it generates train the best predictors, whereas the per-table models, which are trained on the table, fit its support more tightly.

Imputation. The margin is widest here: an Elo of 1451, against 1200 for TabICLv2 and 1093 for tuned ReMasker, and a mean error of 0.129, against 0.146 and 0.159. The gap is largest on categorical cells (0.314, against 0.375 for TabICLv2), where every per-table imputer except TabNAT stays above the mode imputer (0.443).

Class-conditional generation. The same model answers this task by observing the label cell. It ranks first by Elo (1165), ahead of TabPFN-3 (1138) and TabNAT (1124), and its rows are about as separable by class as real rows (label consistency 0.983 of the real one), whereas TabPFGen over-separates them (1.093).

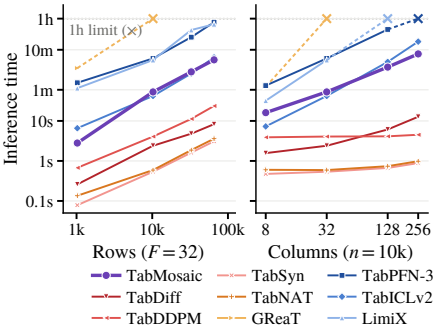

Figure 4: Time to generate one synthetic table against the rows and of columns (median of three runs). ×: over the one-hour limit. Reds: per-table diffusion; oranges: per-table autoregressive; blues: in-context predictors sampled column by column.

Inference time. Fig. 4 reports the time to generate one synthetic table as the rows and the columns grow. TabMosaic takes 53 s at 10,240 rows and 32 columns, against 6–7 min for TabPFN-3 and LimiX. The gap widens with the columns, since the in-context predictors sample one column at a time and re-encode the context for each, whereas TabMosaic denoises all target cells jointly. At 256 columns, TabMosaic takes 8 min, TabICLv2 16 min, and TabPFN-3 and LimiX exceed one hour. The per-table models sample in seconds once trained, but pay the training cost on every table. GReaT exceeds the limit from 10,240 rows.

5.3 RQ2. Posterior predictive fidelity

Corollary 1 states what the objective targets. Following Müller et al. (2022), we test it on priors whose posterior predictive is available in closed form: a Gaussian Process (GP), a GP with a Dirichlet-distributed categorical column, and a Dirichlet prior over the joint of four categorical columns (See Appendix I). Every panel carries the Monte-Carlo floor, i.e., the statistic between two sets of M exact samples, and Fig. 5 adds a factorized oracle, i.e., the product of the exact per-cell posteriors, which bounds any method that fills the cells independently.

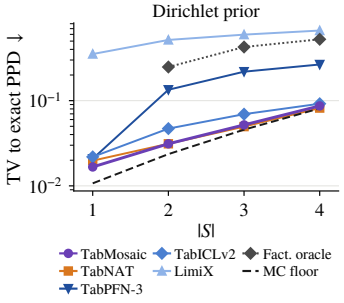

Figure 5: Distance to the exact posterior predictive as the target set S grows on the Dirichlet prior.(Appendix I).

The objective reaches the posterior predictive. We pretrain a small model with our objective Eq. 6 on a mixture of these priors and evaluate it. As shown in the Fig. 3, the NLL approaches that of the exact posterior predictive on every prior, e.g., on the GP at $n = 1,024$, it moves from -0.53 to -0.71 against -0.77 for the exact

Table 1: Error on hidden cells relative to a mean/mode imputer (below 1 means better than mean/mode), for the four best of nine methods on the 24 tables that all nine methods completed. All methods in Table 8.

		TabMosaic	TabICLv2	TabNAT	DiffPuter
MCAR rate	10%	**0.622**	0.729	0.777	0.901
	30%	**0.675**	0.812	0.860	0.966
	50%	**0.770**	0.921	0.932	0.955
	70%	**0.843**	0.974	0.993	1.005
30%	MAR	**0.628**	0.807	0.887	0.887
	MNAR	**0.704**	0.824	0.871	0.866
columns	$\|S\|$=1	**0.523**	0.575	0.780	0.826
	$\|S\|$=2	**0.600**	0.776	0.895	0.822
	$\|S\|$=4	**0.767**	0.888	0.960	0.912

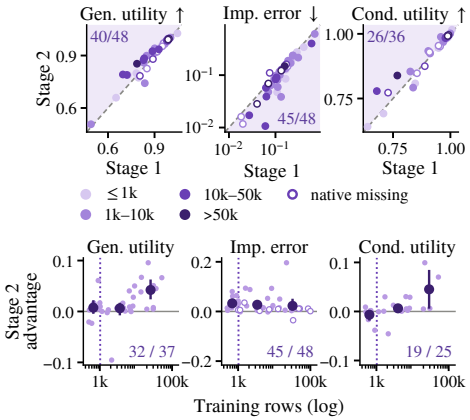

Figure 6: Stage 1 vs. Stage 2: per-table scores (top) and the Stage 2 advantage against table size (bottom). Native missing cells are pre-filled for Stage 1.

one, and on the categorical cells of the GP+Dirichlet prior it reaches 0.114 against 0.103. The same holds beyond the context sizes seen in training ($n = 2048$).

The joint over a target set. We then apply the pretrained TabMosaic, which has never seen this prior, to hidden sets $|S| \in \{1, \ldots, 4\}$, i.e., from imputation to generation (Fig. 5). The factorized oracle degrades as $|S|$ grows, i.e., the dependence among the hidden cells matters. TabMosaic stays within 1.0–1.3 times the floor for $|S| \geq 2$ without training, on par with TabNAT, which is trained on each context table. TabICLv2 is within 1.1–2.0, and TabPFN-3 and LimiX are farther (3.3–5.7 and 8–22, against 6.5–10.5 for the oracle). TabMosaic obtains this joint in one pass without an ordering, whereas the in-context predictors assemble it column by column (Fig. 4).

5.4 RQ3. Target-set performance

Algorithm 1 takes any target set without adaptation. We vary the target set along three axes on real tables, with the training split as context and the test rows as queries (the out-of-sample setting of Zhang et al. 2025a): the *missing rate* $\{10, 30, 50, 70\%\}$, the *mechanism* that selects the hidden cells (MCAR, MAR, MNAR); and the *pattern*, random cells against $|S| \in \{1, 2, 4\}$ whole columns hidden in every query row. Table 1 reports the error on the hidden cells relative to a mean/mode imputer. TabMosaic has the lowest error in every setting. Its error grows with the missing rate (0.62 to 0.84), where the baselines fall back to the imputer (≈ 1), and is flat across the mechanisms.

5.5 RQ4. Staged pretraining

We compare the model after Stage 1 with the final one on the same tables and tasks to isolate what Stage 2 adds. Fig. 6 shows that Stage 2 lowers the imputation error on 45 of the 48 tables, from a mean of 0.156 to 0.129, including 10 of the 11 tables whose context has missing values, which Stage 1 reads only after a mean/mode pre-fill. Generation and class-conditional utility improve on 40 of 48 and 26 of 36 tables, with the gain concentrated on tables above 10k rows, i.e., the lengths that Stage 1 never saw.

6 Conclusion

We posed tabular generation, imputation, and class-conditional generation as one problem, learning the joint conditional distribution over an arbitrary set of target cells on an unseen table, and showed that a diffusion objective over the target set is the natural fit: it defines a valid joint for every mixed-type target set without an ordering, upper-bounds the conditional NLL, and, under the pretraining prior, is minimized by the posterior predictive distribution. TabMosaic, a single model pretrained with this objective in two stages, ranks first on all three tasks across 48 datasets against 20 baselines, approaches the exact posterior predictive on synthetic priors, and serves arbitrary target sets as is.

AI USE STATEMENT

We used generative AI tools to discuss the experimental design, to write and run the code for experiments and figures, to check the proofs in Appendix A, and to draft and edit parts of the text. The research questions, the method, and all decisions originated with the authors, who reviewed every AI-assisted output and take full responsibility for the content.

REFERENCES

Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela Van Der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International conference on machine learning*, pp. 290–306. PMLR, 2022.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022a.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*, 2022b.

Tianyu Du, Luca Melis, and Ting Wang. Remasker: Imputing tabular data with masked autoencoding. In *International Conference on Learning Representations*, volume 2024, pp. 9002–9024, 2024.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *Advances in Neural Information Processing Systems*, 38, 2026.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Noah Hollmann, Samuel Müller, Lennart Purucker, et al. Accurate predictions on small data with a tabular foundation model. *Nature*, 2025.

Daniel Jarrett, Bogdan C Cebere, Tennison Liu, Alicia Curth, and Mihaela van der Schaar. Hyperimpute: Generalized iterative imputation with automatic model selection. In *International conference on machine learning*, pp. 9916–9937. PMLR, 2022.

Xiangjian Jiang, Mingxuan Liu, Nikola Simidjievski, Tassilo Klein, and Mateja Jamnik. Tabular foundation model for generative modelling. *arXiv preprint arXiv:2605.09424*, 2026.

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*, pp. 1288–1296. PMLR, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Jayoung Kim, Chaejeong Lee, and Noseong Park. STaSy: Score-based tabular data synthesis. In *ICLR*, 2023.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *ICML*, 2023.

Xiaofeng Lin, Chenheng Xu, Matthew Yang, and Guang Cheng. Ctsyn: A foundation model for cross tabular data generation. In *International Conference on Learning Representations*, volume 2025, pp. 47938–47959, 2025.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen: Tabular data generation with tabPFN. *arXiv:2406.05216*, 2024.

Andrei Margeloiu, Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. TabEBM: A tabular data augmentation method with distinct class-specific energy-based models. In *NeurIPS*, 2024.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *ICLR*, 2022.

Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the bayesian prism. In *International Conference on Learning Representations*, volume 2024, pp. 49789–49843, 2024.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.

Jingang Qu, David HolzmÃžller, GaÃl Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *Advances in neural information processing systems*, 36:14228–14246, 2023.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, et al. Simple and effective masked diffusion language models. In *NeurIPS*, 2024.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. TabDiff: a mixed-type diffusion model for tabular data generation. In *ICLR*, 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Boris Van Breugel and Mihaela Van Der Schaar. Why tabular foundation models should be a research priority. *arXiv preprint arXiv:2405.01147*, 2024.

Stef Van Buuren and Karin Oudshoorn. Flexible multivariate imputation by mice. *Leiden, The Netherlands: TNO Prevention Center*, pp. 1–20, 1999.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In *ICML*, 2025.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pp. 5689–5698. PMLR, 2018.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, et al. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *ICLR*, 2024.

Hengrui Zhang, Liancheng Fang, Qitian Wu, and S Yu Philip. Tabnat: A continuous-discrete joint generative framework for tabular data. In *ICML*, 2025a.

Hengrui Zhang, Liancheng Fang, Qitian Wu, and Philip Yu. Diffputer: Empowering diffusion models for missing data imputation. In *International Conference on Learning Representations*, volume 2025, pp. 63164–63185, 2025b.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025c.

APPENDIX

TABLE OF CONTENTS

A Proofs **13**

A.1 Proof of Proposition 1 (validity) 13

A.2 Proof of Theorem 1 (conditional ELBO) 13

A.3 Proof of Corollary 1 (the objective targets the posterior predictive distribution) . . . 14

A.4 Proof of Corollary 2 (The posterior predictive minimizes the objective) 14

B Kernel details: EDM and GIDD **14**

C Multi-corruption estimator of the objective **15**

D Missingness in the in-context set **15**

E Pretraining details **16**

F Inference details **16**

G Evaluation protocol **17**

H Generation and imputation with in-context predictors **19**

I Distance to the posterior predictive: protocol **19**

J Target-set scenarios: protocol **20**

K What the second stage adds **20**

L Additional results **21**

A PROOFS

Throughout the appendix, we write $c = (x_{\text{obs}}, D)$ for the conditioning information, $S = S_{\text{num}} \cup S_{\text{cat}}$ for the target set, and $\mathcal{Z} = \mathbb{R}^{|S_{\text{num}}|} \times \prod_{j \in S_{\text{cat}}} \mathcal{V}_j$ for the mixed state space, where $\mathcal{V}_j$ is the class set of column j augmented with the MASK token.

A.1 PROOF OF PROPOSITION 1 (VALIDITY)

Proof. Each factor of the reverse step in Eq. 4 is a normalized Gaussian density or a normalized categorical PMF in its own cell variable, for *any* value of the conditioning arguments (z_t, c, t) and any parameter θ; hence Eq. 4 is a product probability measure on $\mathcal{Z}$ with respect to the product of the Lebesgue and counting measures, i.e., a Markov kernel. The terminal law $p(z_{t_N})$ is a fixed normalized product law, i.e., $\mathcal{N}(0, \sigma_{\max}^2)$ per numerical cell and the MASK point mass per categorical cell. A finite composition of Markov kernels applied to a normalized initial law defines a normalized joint law of $(z_{t_N}, \dots, z_{t_0})$, and marginalizing the intermediate states leaves a normalized law of $z_{t_0} = x_S$, which is Eq. 5. □

We emphasize the two distinct roles of the product structure. The *loss* decomposes additively over cells because the two *step* distributions, Eq. 3 and Eq. 4, factorize given z_t; the *model*, Eq. 5, does not factorize across cells or modalities at the endpoint, because the factors of consecutive steps are coupled through the full mixed state.

A.2 PROOF OF THEOREM 1 (CONDITIONAL ELBO)

Proof. Fix (x, S, c) and the time grid $1 = t_N > \cdots > t_0 = 0$. Writing q for the forward law of $(z_{t_1}, \dots, z_{t_N})$ given x_S and applying Jensen's inequality to Eq. 5 — the only inequality in the proof —

$$-\log p_\theta(x_S \mid c) \;\le\; \mathbb{E}_q\left[-\log \frac{p(z_{t_N}) \prod_{i=N}^{1} p_\theta(z_{t_{i-1}} \mid z_{t_i}, c)}{q(z_{t_1}, \dots, z_{t_N} \mid x_S)}\right], \tag{9}$$

which rearranges into the standard decomposition

$$\sum_{i=2}^{N} \mathbb{E}\,\mathrm{KL}\big(q(z_{t_{i-1}} \mid z_{t_i}, x_S) \,\big\|\, p_\theta(z_{t_{i-1}} \mid z_{t_i}, c)\big) + \mathbb{E}\,\mathrm{KL}\big(q(z_{t_N} \mid x_S) \,\big\|\, p(z_{t_N})\big) + \mathbb{E}\big[-\log p_\theta(x_S \mid z_{t_1}, c)\big]. \tag{10}$$

The prior term is θ-free and vanishes as $\sigma_{\max}$ grows (numerical) and as the terminal mixing completes (categorical); together with the discretization residue of the reconstruction term, it forms the constant C.

For each step term, *both* arguments of the KL factorize over cells: the forward posterior by the cell-wise independence of Eq. 3, and the model by the definition of the step, Eq. 4. The KL divergence between two product measures is the sum of the per-factor KL divergences, so every step term is a sum of per-cell terms, which we group by modality.

For a numerical cell j, both factors are Gaussians with the same variance, differing only in the location parameter, which is a linear function of the clean value; the per-step KL is therefore a quadratic $\kappa_i \, (\hat{x}_{\theta,j}(z_{t_i}, c; t_i) - x_j)^2$ with a θ-free coefficient κ_i determined by the schedule. Summing over the steps and passing to the continuous-time limit yields $\mathbb{E}_t \, w_{\text{lik}}(t) \, (\hat{x}_{\theta,j} - x_j)^2$ with the likelihood weight $w_{\text{lik}}(t) = \sigma'(t)/\sigma(t)^3$; with this weight, the numerical part of the bound is exact up to discretization. Replacing w_{lik} by the EDM weight $\lambda(\sigma(t))$ (Appendix B) multiplies each (t, z_t)-indexed term by a positive factor. Since the minimization over $\hat{x}_{\theta,j}(z_t, c; t)$ is pointwise in (t, z_t), every positive weighting has the same minimizer $\hat{x}_{\theta,j} = \mathbb{E}[x_j \mid z_t, c]$; the reweighting changes the optimization geometry, not the target.

For a categorical cell j, the per-step KL divergences under the GIDD kernel telescope, in the continuous-time limit, into the exact weighted form $\mathbb{E}_t \, w(z_{t,j}) \, [D_{\text{KL},j} + D_{\text{IS},j}]$ of von Rütte et al. (2025), where the weights and both divergence terms are available in closed form (Appendix B). Collecting the numerical cells into $\mathcal{L}_{\text{num}}$ and the categorical cells into $\mathcal{L}_{\text{cat}}$ gives the statement. □

A.3 Proof of Corollary 1 (the objective targets the posterior predictive distribution)

Proof. Let tables be drawn from the prior, $(x \cup D) \sim p(\mathcal{D})$, with the target pattern drawn independently of the data, $S \perp (x, D)$. Taking expectations of the pointwise bound of Theorem 1,

$$\mathbb{E}\big[\ell_\theta^\star\big] + C \;\geq\; \mathbb{E}\big[-\log p_\theta(x_S \mid x_{\text{obs}}, D)\big]. \tag{11}$$

Conditionally on (S, x_{obs}, D), the clean target x_S is distributed as the posterior predictive $p(x_S \mid x_{\text{obs}}, D)$ of Eq. 2, because the query row and the context rows are i.i.d. given the mechanism ϕ. The cross-entropy decomposition then gives

$$\mathbb{E}\big[-\log p_\theta(x_S \mid x_{\text{obs}}, D)\big] = \mathbb{E}\big[H\big(p(\cdot \mid x_{\text{obs}}, D)\big)\big] + \mathbb{E}\big[\mathrm{KL}\big(p(\cdot \mid x_{\text{obs}}, D) \,\big\|\, p_\theta(\cdot \mid x_{\text{obs}}, D)\big)\big], \tag{12}$$

where the entropy term is independent of θ; the constant C' of the statement collects C and this entropy. Minimizing the loss therefore minimizes an upper bound on the expected KL divergence to the posterior predictive. That $\mathcal{L}$ and $\mathbb{E}[\ell_\theta^\star]$ share their minimizer follows from the proof of Theorem 1: the two differ only in the positive per-(t, z_t) weights of the numerical terms. When S is a single cell and the model class is an explicit classifier, the diffusion ELBO is replaced by the exact log-likelihood and the bound collapses to the cross-entropy identity of §2.1; the corollary is the same statement with the single cell generalized to an arbitrary mixed-type cell set and the equality relaxed by the ELBO. □

A.4 Proof of Corollary 2 (The posterior predictive minimizes the objective)

Proof of Corollary 2. Under the assumptions, (x_S, z_t) in Eq. 6 is distributed as $p(x_S \mid c)\, q_t(z_t \mid x_S)$. Conditioned on (z_t, c), the numerical term is a squared loss, minimized by the posterior mean, and the categorical term is the GIDD ELBO term, which is minimized by the true posterior PMF (von Rütte et al., 2025). Both minimizers are pointwise in (z_t, c, t) and independent of the positive weights, hence they minimize $\mathcal{L}$ and $\mathbb{E}[\ell_\theta^\star]$ alike. With this denoiser every factor of Eq. 4 is the true reverse transition of the forward process started from $p(x_S \mid c)$, and as $N \to \infty$ the per-step KL terms of the ELBO vanish (Song et al., 2021; von Rütte et al., 2025), so the composition has marginal $p(x_S \mid c)$ at $t = 0$. □

B Kernel details: EDM and GIDD

Shared time. One time $t \sim \mathcal{U}[\epsilon, 1-\epsilon]$ with $\epsilon = 10^{-3}$ is drawn per table and shared by the numerical and the categorical cells of its target set. The numerical noise level follows the EDM schedule $\sigma(t) = \big(\sigma_{\min}^{1/\rho} + t\,(\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho})\big)^\rho$ with $\sigma_{\min} = 0.002$, $\sigma_{\max} = 80$, and $\rho = 7$ (Karras et al., 2022); the categorical mixing coefficients are those of the GIDD schedule below at the same t.

Numerical cells (EDM). Numerical cells follow Karras et al. (2022): the forward kernel is $q_\sigma(z \mid x) = \mathcal{N}(z; x, \sigma^2)$ and the denoiser uses the EDM preconditioning with data scale $\sigma_d = 1$, since every numerical column is mapped to a standard normal marginal by a per-column quantile transform fit on the observed context cells (Appendix E). The training weight is the EDM weight $\lambda(\sigma) = (\sigma^2 + \sigma_d^2)/(\sigma\sigma_d)^2$, which equalizes the gradient magnitude across noise levels and, as shown in Appendix A.2, shares its minimizer $\mathbb{E}[x \mid z, c]$ with the likelihood weight at every (z, σ).

Categorical cells (GIDD). Categorical cells follow the interpolating family of von Rütte et al. (2025), whose forward marginal is Eq. 3 with a mixing distribution π_t that blends the MASK point mass with uniform noise; we use their mixing rate $p_u = 0.1$ and $\gamma = 1$, under which the uniform component vanishes at both ends, so that $\alpha_0 = 1$ (clean) and the terminal law is exactly the MASK point mass. The per-cell loss carries the exact continuous-time ELBO weights of their Theorem 3.7, $\mathbb{E}_t\, w(z_{t,j})\,[D_{\mathrm{KL},j} + D_{\mathrm{IS},j}]$, where $D_{\mathrm{KL},j}$ compares the true and the model one-step posteriors, $D_{\mathrm{IS},j}$ is the importance-sampled correction for non-MASK intermediate states, and the weight w is available in closed form. Three properties matter for our construction: i) the weights are exact, so the categorical block enters Theorem 1 without any reweighting caveat; ii) unlike masked (absorbing) diffusion (Austin et al., 2021; Sahoo et al., 2024), where an unmasked token can never change again,

the uniform component lets the reverse process revise an already-decoded token; and iii) MASK is a legal state at every t and carries no information about the clean value, which is what lets a categorical target cell enter the reverse process at any t (Appendix F).

C MULTI-CORRUPTION ESTIMATOR OF THE OBJECTIVE

Theorem 1 and Corollary 1 concern the population objective

$$\mathcal{J}(\theta) = \mathbb{E}_{(x\cup D)\sim p(\mathcal{D})}\, \mathbb{E}_S\, \mathbb{E}_t\, \mathbb{E}_{z_t\sim q_t(\cdot|x_S)}\big[\ell_\theta(z_t, t; x_S, c)\big], \qquad \ell_\theta = \tfrac{1}{|S|}\big[\mathcal{L}_{\text{num}} + \mathcal{L}_{\text{cat}}\big], \tag{13}$$

which is $\mathcal{L}$ of Eq. 6. Training replaces the inner expectations by samples. Per optimization step, we draw G tables; for table g, we draw *one* time t_g and *one* target pattern S_g, and then K independent corruption realizations of the same (t_g, S_g):

$$z^{(g,k)} \overset{\text{iid}}{\sim} q_{t_g}\big(\cdot \mid x_{S_g}\big), \qquad \widehat{\mathcal{J}}_{G,K}(\theta) = \frac{1}{G}\sum_{g=1}^{G}\frac{1}{K}\sum_{k=1}^{K}\ell_\theta\big(z^{(g,k)}, t_g;\; x_{S_g}, c_g\big), \tag{14}$$

whose inner average is $\hat{\ell}_K$ of Eq. 8. $K = 1$ is the standard single-corruption estimator; $K > 1$ averages the *losses* of the K copies, not their outputs.

Proposition 2 (Unbiasedness and variance)**.** *For every $K \geq 1$, $\mathbb{E}[\widehat{\mathcal{J}}_{G,K}] = \mathcal{J}$ and $\mathbb{E}[\nabla_\theta \widehat{\mathcal{J}}_{G,K}] = \nabla_\theta \mathcal{J}$, and*

$$\operatorname{Var}\big(\nabla_\theta \widehat{\mathcal{J}}_{G,K}\big) = \frac{1}{G}\left(V_{\text{out}} + \frac{V_{\text{in}}}{K}\right), \qquad \begin{aligned} V_{\text{out}} &= \operatorname{Var}_{(x,D,S,t)}\big(\mathbb{E}_z[\nabla_\theta \ell_\theta]\big), \\ V_{\text{in}} &= \mathbb{E}_{(x,D,S,t)}\big[\operatorname{Var}_z(\nabla_\theta \ell_\theta)\big]. \end{aligned} \tag{15}$$

Hence K reduces exactly the corruption-conditional component V_{in} by $1/K$, while G reduces both, and the minimizer is unchanged by K.

Proof. Each summand of Eq. 14 is an unbiased sample of Eq. 13 by construction, which gives unbiasedness, and, under standard regularity, the same for the gradient. The variance follows from the law of total variance conditioned on (x, D, S, t): the inner mean of K i.i.d. draws has conditional variance $\operatorname{Var}_z(\nabla_\theta \ell_\theta)/K$, and the G tables are independent. □

D MISSINGNESS IN THE IN-CONTEXT SET

The second pretraining stage additionally corrupts the *conditioning* side: some cells of the context rows are hidden from the model. This changes the conditioning object, not the bound.

Masking distribution. Let $R \in \{0,1\}^{|D|\times F}$ be an observation mask over context cells. For each table we draw a rate $\rho \sim \mathcal{U}(0, \rho_{\max}(s))$, where s is the training step and $\rho_{\max}(s) = 0.2\min(1, s/2{,}500)$; a pattern, i.e., i.i.d. cells (probability 0.5), a random subset of columns (0.3) or of rows (0.2) hidden at a higher per-cell rate $q \sim \mathcal{U}(0.5, 1)$ and included with probability ρ/q, so that the expected fraction of hidden cells is ρ under every pattern; and a mechanism, i.e., MCAR (probability 0.5), MAR (0.25) or MNAR (0.25). Under MAR, the per-cell rate of a row is tilted by the within-column rank of one randomly chosen driver column, which stays fully observed; under MNAR, the rate of a cell is tilted by the rank of its own value (self-masking). The tilt amplitude is drawn uniformly in $[0, 2\min(\rho, 1-\rho)]$, so the expected rate remains ρ and every cell probability stays in $[0, 1]$. The label column is masked like any other column. The masked context and the conditioning object are

$$\widetilde{D}_{ij} = \begin{cases} D_{ij}, & R_{ij} = 1, \\ \emptyset, & R_{ij} = 0, \end{cases} \qquad \widetilde{c} = \big(x_{\text{obs}}, \widetilde{D}, R\big), \tag{16}$$

where $\emptyset$ is realized in the network as the MASK token (categorical) or the value 0 (numerical) *together with* the third observation state in $\{\text{target}, \text{observed}, \text{missing-in-context}\}$, so that the network reads both the observed values and the pattern R. The per-column quantile transform of the numerical columns is fit on the observed context cells only, so $\widetilde{c}$ is a function of the observed part of D and of R. Missing cells of the *query* row need no new state: at inference, they are precisely the cells to be imputed, i.e., they belong to S. The training objective is Eq. 13 with c replaced by $\widetilde{c}$; we denote it $\mathcal{J}_s(\theta)$.

E PRETRAINING DETAILS

Synthetic prior. Tables are drawn from the structural-causal-model prior of TabICL v2 (Qu et al., 2026) with the two changes of §4.2: the label is an ordinary column whose cardinality follows the same law as the categorical features (up to 200 classes), and half of the tables have a numerical label. We use the corrected sampling of categorical cardinalities of its implementation, graphs with 2 to 32 nodes, and its filters that discard graphs and tables whose label is unpredictable from the features. Every numerical column is mapped to a standard normal marginal by a per-column quantile transform fit on the observed context cells; values are mapped back by the inverse transform at inference.

Target pattern. For each table, with probability 0.4 all cells of the query rows are targets (generation); otherwise a rate $\rho \sim \mathcal{U}(0, 1)$ is drawn and each query cell is a target independently with probability ρ. The pattern is drawn independently of the cell values and of t (§3.3).

Table shapes. Stage 1 uses $T = 1{,}024$ rows per table, $F \sim \log\mathcal{U}(1, 100)$ features, and a context fraction drawn from $\mathcal{U}(0.3, 0.9)$; tables have no missing values. Stage 2 draws the number of rows first, $T \sim \log\mathcal{U}(400, 10{,}240)$, and then the number of features, $F \sim \log\mathcal{U}(4, \min(200, \lfloor B/T \rfloor))$, with the cell budget $B = 250{,}000$, so that $T \times F \leq B$; the context fraction is 0.8. Tables in a micro-batch of four share one shape.

Table 2: Pretraining configuration. Both stages use 8 GPUs.

	Stage 1	Stage 2
Rows T	1,024	$\log\mathcal{U}(400, 10{,}240)$
Features F	$\log\mathcal{U}(1, 100)$	$\log\mathcal{U}(4, \min(200, B/T))$, $B = 250{,}000$
Context fraction	$\mathcal{U}(0.3, 0.9)$	0.8
Context missingness	—	$\rho \sim \mathcal{U}(0, \rho_{\max})$, $\rho_{\max}$: $0 \to 0.2$ over 2,500 steps
Corruptions K per table	1	4
Steps / tables seen	125k / 32M	10k / 2.56M
Global batch (tables)	256	256
Peak learning rate	8×10^{-4}	10^{-4}
Warm-up	1%	2%
Initialization	random	stage-1 final

Stages. Table 2 summarizes the two-stage recipe. Stage 2 warm-starts from the final stage-1 checkpoint and introduces the long tables, the context-missingness distribution of Appendix D, and the multi-corruption estimator of Appendix C with $K = 4$. The evaluated model is the stage-2 checkpoint at step $10{,}000$; no per-table training, tuning, or selection is applied anywhere in the paper.

Architecture and optimization. The denoiser is the column–row attention transformer of §4.1 with 12 blocks, width 384, 8 heads, a feed-forward expansion of 2, pre-normalization, and normalized queries and keys (29M parameters). Each cell token combines a type-specific value embedding (a linear map for numerical cells; a class embedding with a shared vocabulary of 201 entries, the last of which is MASK, for categorical cells), the three-state observation embedding, and, for target cells only, the embedding of t. Optimization uses Muon for the two-dimensional weights and AdamW for the remaining parameters at $0.2\times$ the learning rate, with $\beta_1 = 0.9$, weight decay 0.01, a cosine schedule with linear warm-up decaying to 10^{-7}, gradient clipping at 10, bf16 precision, and activation checkpointing.

F INFERENCE DETAILS

All the tasks run Algorithm 1 with the same network; Table 3 lists the settings. The context is the training split of the table, capped at $8{,}192$ rows: a random subset for generation and a label-stratified random subset for imputation. Cells of the context that are missing in the table are presented in the missing-in-context state (Appendix D); the quantile transform is fit on the observed cells.

Table 3: Settings per task. t_{start} is the time at which the reverse process starts, N the number of steps, and M the number of trajectories per query row; the context is encoded once per query batch.

Task	S	x_{obs}	t_{start}	Sampler (num.) / N	M; output
Generation	all query cells	—	1	deterministic / 128	1; one row per trajectory
Class-cond. gen.	feature cells	label cell	1	deterministic / 128	1; one row per trajectory
Imputation	missing cells	observed cells	0.5	ancestral / 16	8; mean (num.), vote (cat.)

Starting the reverse process at $t_{\text{start}} < 1$. Since $S \perp t$ during pretraining, the reverse process can be entered at any $t_{\text{start}} \leq 1$. Categorical target cells enter at MASK, which lies in the support of the forward marginal at every t and carries no information about the clean value (Appendix B). Numerical target cells enter at $z_{t_{\text{start}}} = \hat{x}^{(0)} + \sigma(t_{\text{start}})\,\varepsilon$, where $\hat{x}^{(0)} = \hat{x}_\theta(\sigma_{\max}\varepsilon'; \sigma_{\max}, c)$ is the output of one denoiser evaluation at the terminal noise level, i.e., the conditional mean. Imputation uses $t_{\text{start}} = 0.5$ with $N = 16$ ancestral steps, which we selected on a development checkpoint: per-cell error metrics saturate at a handful of steps, whereas the distribution-level metrics of generation prefer the full trajectory, so generation starts at $t_{\text{start}} = 1$ with $N = 128$.

In other words, at $N{=}1$ a numerical target cell returns the conditional mean and a categorical target cell is drawn from the denoiser PMF, each cell independently of the others; this is property iii) of §3.2. The coupling between the cells of S is created by the iteration: with $N > 1$ every step reads the current state of every other target cell (Eq. 4).

Point estimates from samples. Imputation is scored by per-cell error metrics (Appendix G), which require one value per cell. We take the mean over the $M = 8$ trajectories for a numerical cell and the majority vote for a categorical cell (ties are broken by the mode of the observed cells of the column, then the lowest class index). Distribution-level metrics use the trajectories as they are, one sample per trajectory.

Generation. Generation and class-conditional generation run the full reverse process from the terminal law over all the target cells jointly; for class-conditional generation the label cell of every query row is observed and set to the requested class. The denoiser PMF of a categorical cell is used at temperature 1: lowering it trades diversity for marginal sharpness and degrades distribution-level metrics. Each evaluation seed regenerates the entire table, i.e., seeds are whole-table replicates.

Cost. The context is encoded once per query batch and reused across steps, query rows, and trajectories (§4.3); query rows are processed in blocks of 8,192. The median wall-clock of generation is 25 s per table on one GPU, whereas the per-table generators train on every table, e.g., TabNAT for a median of 860 s per table over the 48 tables and three seeds; the scaling with rows and columns is in Fig. 4.

G Evaluation protocol

Benchmark. All the evaluations use the TabArena suite (Erickson et al., 2026) under its single-split (lite) protocol, i.e., the first official outer split of each dataset. Three of the 51 tables have more than 1,000 features; we exclude them from every task and every method, which leaves 48 tables with 498 to 100,000 training rows and up to 212 features. Class-conditional generation uses the 36 classification tables among them. We report Elo, ranks, and means without statistical significance claims. In the per-size analyses, tables are grouped by training rows into $\leq$ 1k (12 tables), 1k–10k (21), 10k–50k (11), and $>$ 50k (4).

Scoring predictors. Utility and label consistency are measured with three predictors — LightGBM, CatBoost, and TabPFN — at their default configurations; the score of a table is the mean over the three. We use three predictors so that no single inductive bias decides the ranking; the predictors are never tuned on synthetic data.

Generation. Every generator is given the official training split as its training data (per-table models) or as its context (TabMosaic and the in-context generators) and synthesizes n_{train} rows. Scores are i) *utility*: the official metric of the scoring predictors trained on synthetic rows and tested on the real test split, divided by the same metric when trained on the real training rows (the ratio is inverted for regression metrics where lower is better); ii) α-precision and β-recall (Alaa et al., 2022); and iii) authenticity (Alaa et al., 2022). Column-level statistics — Wasserstein-1 on min-max–scaled numerical columns, total variation on categorical columns, and the mean absolute difference of pairwise correlations — are reported as diagnostics (Appendix L). The pool comprises i) per-table generators (Kotelnikov et al., 2023; Kim et al., 2023; Zhang et al., 2024; Shi et al., 2025; Zhang et al., 2025a; Jolicoeur-Martineau et al., 2024; Borisov et al., 2022b), tuned over 10 configurations from their published search spaces where their paper tunes and run with their published configuration otherwise (TabNAT, ForestDiffusion, and GReaT publish no search space and are run default-only); ii) two reference baselines, a Gaussian mixture over the row and independent column resampling, which fix the scale of the Elo; and iii) in-context generators that chain a pretrained predictor over the columns without training on the table (Hollmann et al., 2025; Qu et al., 2025; Zhang et al., 2025c), run with their official generation procedures (Appendix H). Every stochastic method is run with 3 seeds; each seed regenerates the whole table, scores are computed per seed, and the *scores* are averaged. On classification tables, TabDDPM is trained with the label as its conditioning input and samples labels from the empirical class proportions, and CTGAN samples its discrete-column condition vector from the data frequencies; on regression tables both treat the target as a numerical feature, as in their papers.

Imputation. Missing cells are injected into the feature cells of the training split completely at random at a 30% rate with 3 mask seeds; the label column is never injected and is passed to every imputer as an ordinary observed column; natural missingness is passed as missing, excluded from injection, and never scored. Each imputer fits on the incomplete table and fills it, and only the injected cells are scored: min-max–scaled RMSE on numerical cells and the misclassification rate on categorical cells, averaged per table; the primary score of a table is the RMSE when the table has numerical cells and the misclassification rate otherwise. The pool comprises a column mean/mode imputer (the reference baseline), MICE (Van Buuren & Oudshoorn, 1999), deep imputers (Yoon et al., 2018; Jarrett et al., 2022; Du et al., 2024; Zhang et al., 2025b), the two per-table generators whose papers give an imputation procedure (Zhang et al., 2025a; Jolicoeur-Martineau et al., 2024), and in-context imputers that fill each missing cell from the observed ones (Hollmann et al., 2025; Qu et al., 2025; Zhang et al., 2025c); methods whose paper tunes are also run tuned (marked (t)). TabMosaic takes the incomplete table as context, with its missing cells in the missing-in-context state, and answers every row with one conditional query over its missing cells (Appendix F).

Class-conditional generation. The condition of a table is its training-split label vector, so the synthetic table has n_{train} rows with the real class proportions. Scores are utility (as above), label consistency — the fraction of synthetic rows that the scoring predictors, trained on the real training rows, assign to the requested class, reported relative to the same fraction on held-out real rows — and the per-class Wasserstein-1 and total variation, macro-averaged over classes. Label agreement, the fraction of rows whose emitted label equals the requested class, is 1 for every method except CTGAN, whose condition is a soft constraint; we score CTGAN's output as emitted and do not apply rejection sampling, which would be a procedure outside the model. Table 4 lists how each method fixes the condition.

Elo. We follow the official TabArena procedure: every (table, metric) pair is one task, a method's score on a task is its rank among the entered methods, and Elo ratings are fit to the pairwise win rates with 200 bootstrap resamples over tasks for the 95% intervals. Generation tasks are table $\times$ {utility, α-precision, β-recall, authenticity}; imputation tasks are table $\times$ {RMSE on numerical cells, error rate on categorical cells, downstream utility}, where a table enters only the metrics its cells admit; class-conditional tasks are table $\times$ {utility, class-macro α-precision, class-macro β-recall, label-consistency gap}. A method that has not been run on a table is entered with the reference baseline's score on that table and marked † in the figures; every such fill is disclosed with its count.

Excluded methods. TabSyn and TabDiff are not entered for imputation or class-conditional generation, because their papers claim neither task in the form we evaluate: TabSyn's paper evaluates

Table 4: How each method conditions on the class in class-conditional generation.

Mechanism	Methods	Where the conditional comes from
Pretrained conditional	TabMosaic	label cell observed; the conditional is a training term
Trained conditional, per table	TabNAT	masked training over random orders on the table
Class input at training	TabDDPM	the label is an input of the denoiser; not generated
One model per class	ForestDiffusion	a separate model is trained for each class
Discrete-column condition	CTGAN	condition vector plus a cross-entropy penalty (soft)
Prompt	GReaT	the label appears in the prompt; permutation training
Classifier energy	TabEBM, TabPFGen	Langevin sampling of a pretrained classifier's energy
Chained given the label	in-context generators	label observed; the columns are chained

unconditional generation and imputation of a target column by inpainting, and TabDiff's imputation experiments impute the target column only and require an auxiliary model trained on that column. TVAE, STaSy, and the reference baselines have no conditioning mechanism. CTSyn and TabFORGE (Lin et al., 2025; Jiang et al., 2026) have no public code at the time of writing and are discussed in §2.2 only.

H Generation and imputation with in-context predictors

The in-context generators and imputers of Appendix G are pretrained on one cell at a time: a label cell (Hollmann et al., 2025; Qu et al., 2025) or masked cells scored separately given the unmasked ones (Zhang et al., 2025c). Their official procedures obtain a row or a set of cells from these single-cell conditionals. i) *Generation* chains the predictor over the columns, $p(x_1 \mid D)\, p(x_2 \mid x_1, D) \cdots$, in one or several column orders and averages over the orders; the chain is a valid joint for each order, but each order is a different joint, and the conditionals were never trained together. LimiX follows the chain with rounds of masking a fraction of the generated cells and refilling them. ii) *Imputation* fills the missing cells of a row from the observed cells, column by column with the cells already filled entering the conditioning (TabPFN-3, TabICLv2) or all at once (LimiX), and returns one value per cell, i.e., a point estimate at the mode or the mean of each conditional rather than a sample of the joint; Appendix I measures the distance of such point estimates from the joint. iii) The masked objective $-\log q_\theta(x_\pi \mid x_{-\pi}, D)$ of LimiX is written over a masked set π, but the network outputs one distribution per masked cell given the unmasked cells, so for $|\pi| > 1$ the objective is the pseudo-likelihood-style product $\prod_{j\in\pi} q_\theta(x_j \mid x_{-\pi}, D)$, which parameterizes no joint over π. In TabMosaic, by contrast, the cells of S are coupled at every reverse step and the objective is a bound on the likelihood of x_S itself (Theorem 1).

I Distance to the posterior predictive: protocol

Corollary 1 and Corollary 2 are statements about the pretraining prior, so we test them as Müller et al. (2022) test theirs, on priors whose posterior predictive is available in closed form. Two experiments use these priors differently. Fig. 3 pretrains a small model on the priors themselves and follows its checkpoints. Fig. 5 applies the pretrained TabMosaic of §5.2, which has never seen these priors, to the Dirichlet prior, so its distances include a prior-mismatch term that the corollaries do not bound.

Priors. x denotes a numerical and c a categorical column. i) *GP*: $x_1, x_2 \sim \mathcal{U}(-2, 2)$ and $x_3 = f(x_1, x_2) + \varepsilon$ with $f \sim \mathcal{GP}(0, k)$, k the RBF kernel with output scale 1, and $\varepsilon \sim \mathcal{N}(0, 0.1^2)$. The posterior predictive of x_3 given the other cells and the context is the GP posterior; the length scale is fixed to 0.6, so that the posterior predictive is exact. ii) *Dirichlet*: four categorical columns $c_1, \ldots, c_4$ with three classes each, one table-level PMF $\theta \sim \mathrm{Dir}(0.3 \cdot \mathbf{1})$ over the 81 joint cells, and rows drawn i.i.d. from θ; the posterior predictive of any subset of cells given the rest is Dirichlet–multinomial. iii) *GP+Dirichlet*: one categorical column c with three classes and class proportions $\pi \sim \mathrm{Dir}(0.3 \cdot \mathbf{1})$, two numerical columns as in i), and $x_3 \sim \mathcal{GP}(0, k)$ with $k = \mathrm{RBF}(x) + \mathbf{1}[c = c']$, so that rows of the same class are correlated. The posterior of x_3 given (x_1, x_2, c) is the GP posterior with the extended kernel, and that of c given (x_1, x_2, x_3) follows by enumerating the three classes.

Small model (Fig. 3). The model is a 6-block, width-256 instance of the architecture of Appendix E (6.7M parameters), trained with the Stage-1 recipe on a mixture of the three priors with equal probability, context length log-uniform in $[8, 1{,}024]$, 16 query rows, and the target pattern of Appendix E, for 60k steps on one GPU; numerical cells are standardized with the mean and the standard deviation of the observed context cells. We evaluate the checkpoints at 10k, 20k, 30k, and 60k steps on 20 tables per prior, 16 query rows per table, context sizes $n \in \{8, \dots, 2{,}048\}$, and $M = 512$ samples per query. The score is the negative log-likelihood of the true value of the hidden cells under the samples: the empirical frequency of the true cell among the samples for categorical targets and a Gaussian kernel density of the samples for a numerical target. The exact posterior predictive is scored in closed form, and the floor applies the same estimator to M of its own samples.

Target sets (Fig. 5). Each query hides a set S of cells of a row of the Dirichlet prior, $|S| \in \{1, \dots, 4\}$, with $n = 1{,}024$ context rows, 20 tables, 16 query rows per table, and $M = 1{,}024$ samples per query. The distance to the posterior predictive is the total variation to the exact joint PMF. The figure carries the *Monte-Carlo floor*, the same statistic between the exact posterior predictive and M of its own samples, and the *factorized oracle*, the product of the exact per-cell marginals, which is the best any method that fills the cells of S independently can reach. The baselines sample the same cells with their own generation procedures, conditioned on the observed cells: TabNAT is trained on the context table and samples the hidden cells autoregressively in a random order; TabPFN-3 samples them column by column from its predictive distribution at temperature 1, averaging 10 permutations of the conditioning columns per column; TabICLv2 runs the column-wise chain of its official generation procedure over the hidden columns in the original order at temperature 1; LimiX runs the autoregressive stage of its official procedure over the hidden columns and then three rounds of masking and refilling within S. Their imputation procedures, which return one value per cell, are farther from the posterior predictive than the factorized oracle at every $|S|$ and are omitted from the figure.

J TARGET-SET SCENARIOS: PROTOCOL

Unlike the imputation benchmark of Appendix G, which injects missingness into the training split and fills that split, the scenarios of §5.4 keep the training split as it is, natural missingness included, and hide cells of the *test* rows, so that the context is fixed and only the target set varies; the hidden values are known and scored. Hidden cells are drawn by the same generator as the benchmark (MCAR at rates 10–70%, MAR with logistic drivers among the numerical features, MNAR by self-masking; Zhang et al. 2025b), or by hiding $|S| \in \{1, 2, 4\}$ whole feature columns, chosen at random per table, in every test row. Scores are the per-cell errors of the benchmark, divided by the errors of the mean/mode imputer under the same mask; values below 1 improve on it. The eight baselines are MICE, GAIN, DiffPuter, ForestDiffusion, and TabNAT, each fit on the training split and applied to the test rows, and the in-context imputers TabPFN-3, TabICLv2, and LimiX with the training split as context. TabMosaic runs Algorithm 1 unchanged over the whole grid with the training split as context and the imputation settings of Table 3. All results use one mask seed. TabPFN-3, ForestDiffusion, and DiffPuter did not finish the larger tables within our compute budget, so Table 1 reports the 24 tables on which all nine methods completed, and Table 8 reports every method on those tables together with the number of tables each method has results on.

K WHAT THE SECOND STAGE ADDS

Stage 2 introduces long tables, the context-missingness distribution, and the multi-corruption estimator (Appendix E). We compare the final Stage-1 checkpoint with the final model on the three tasks of Appendix G, with the same inference settings. The Stage-1 model has no state for a missing context cell, so on the 11 tables whose context has natural missingness its context is pre-filled (mean/mode for imputation; median and an explicit missing category for generation), whereas the final model reads the incomplete context directly; on these tables the two models therefore see different inputs, which we mark in Fig. 6 and exclude from the size analysis of generation and class-conditional generation. Imputation error is lower after Stage 2 on 45 of 48 tables (mean 0.129 against 0.156) and on every size group. Generation utility improves on 40 of 48 tables and class-conditional utility on

26 of 36, and both gains concentrate on tables above 10k rows ($+0.04$ on 9 of 9 and 4 of 4 tables), i.e., the lengths that Stage 1 never saw.

L ADDITIONAL RESULTS

Full results. Tables 5, 6, and 7 report every method on generation, imputation, and class-conditional generation over the 48 and 36 tables, including the reference baselines and the in-context methods.

Table 5: Generation on the 48 tables: Elo with 95% bootstrap intervals and per-metric means over the tables each method completed (n). Utility is relative to training on real rows; α-precision, β-recall, and authenticity follow Alaa et al. (2022); W_1 (median over tables) and TV are column-level distances and $|\Delta\rho|$ the mean absolute difference of pairwise correlations. Time is the median wall-clock per table in seconds (training plus sampling for per-table models; inference for the others), measured on one GPU. The Elo tasks are table $\times$ {utility, α-precision, β-recall, authenticity}. (t) = run with the hyperparameters tuned in the original paper, which provides no default. †: not completed on every table; missing tables are filled with column resampling in the Elo.

Method	n	Elo	Utility ↑	α-prec. ↑	β-rec. ↑	Auth. ↑	W_1 ↓	TV ↓	$\lvert\Delta\rho\rvert$ ↓	Time (s)
TabMosaic	48	1104 [1066, 1134]	0.914	0.790	0.408	0.569	0.054	0.038	0.038	25
TabNAT†	42	1070 [1042, 1095]	0.896	0.836	0.439	0.541	0.049	0.020	0.028	860
TabSyn	48	1065 [1034, 1089]	0.864	0.738	0.385	0.590	0.059	0.055	0.037	369
TabDiff (t)†	46	1059 [1034, 1086]	0.885	0.741	0.385	0.584	0.060	0.022	0.041	2003
TabPFN-3 (chained)	48	1058 [1029, 1090]	0.891	0.662	0.353	0.632	0.079	0.081	0.059	571
ForestDiffusion†	36	1034 [1006, 1062]	0.886	0.836	0.442	0.539	0.061	0.048	0.030	712
TabICLv2 (chained)	48	1021 [995, 1047]	0.880	0.590	0.304	0.648	0.169	0.062	0.081	72
TabDDPM (t)†	45	1006 [970, 1047]	0.818	0.660	0.371	0.618	0.263	0.095	0.137	241
TVAE (t)†	47	1000 [983, 1021]	0.795	0.670	0.320	0.628	0.123	0.123	0.081	529
TabEBM†	31	987 [961, 1017]	0.917	0.630	0.397	0.545	0.051	0.062	0.020	155
GReaT†	34	982 [961, 1002]	0.788	0.750	0.393	0.596	0.288	0.145	0.115	1101
TabPFGen†	31	969 [940, 997]	0.851	0.647	0.405	0.549	0.077	0.059	0.042	131
STaSy (t)†	46	965 [932, 990]	0.799	0.542	0.270	0.691	0.192	0.111	0.072	1430
Gaussian mixture†	47	951 [926, 975]	0.777	0.521	0.251	0.689	0.179	0.056	0.028	11
CTGAN (t)†	47	946 [924, 966]	0.729	0.488	0.230	0.711	0.160	0.112	0.113	259
LimiX (chained)†	46	894 [852, 934]	0.649	0.345	0.133	0.830	0.683	0.277	0.153	219
Column resampling	48	890 [862, 917]	0.518	0.425	0.197	0.743	0.017	0.017	0.203	0

Table 6: Imputation on the 48 tables (30% of the feature cells hidden completely at random; 3 mask seeds). The Elo tasks are table × {RMSE on numerical cells, error rate on categorical cells, downstream utility} (125 tasks); the second Elo uses only the 42 tables every method completed (107 tasks). Error is the primary score of a table (min-max–scaled RMSE where the table has numerical cells, misclassification rate otherwise), averaged over the tables each method completed (n); rank is the mean rank over tables. RMSE and error rate are also reported separately over the tables with numerical and with categorical cells (counts in parentheses); cells with RMSE above 5 (out-of-range outputs) are excluded from the means and counted. Downstream is the utility of a predictor trained on the imputed table relative to one trained on the complete table. (t) = tuned. †: not completed on every table; missing tables are filled with the mean/mode imputer in the first Elo.

Method	n	Elo	Elo (common)	Error ↓	Rank ↓	RMSE ↓ (num.)	Error rate ↓ (cat.)	Downstream ↑
TabMosaic	48	1451 [1377, 1515]	1447 [1380, 1532]	0.129	1.71	0.108 (44)	0.314 (33)	0.962
TabICLv2 (per cell)	48	1200 [1163, 1239]	1220 [1175, 1265]	0.146	4.12	0.126 (44)	0.375 (33)	0.839
ReMasker (t)†	46	1093 [1052, 1131]	1105 [1073, 1147]	0.159	3.59	0.122 (42)	0.472 (32)	0.829
HyperImpute	48	1077 [1046, 1103]	1079 [1049, 1111]	0.163	4.85	0.129 (44)	0.473 (33)	0.828
DiffPuter†	47	1068 [1021, 1108]	1084 [1046, 1122]	0.207 (3 excl.)	6.91	0.173 (43)	0.466 (33)	0.907
ReMasker†	39	1039 [1012, 1068]	1050 [1016, 1086]	0.178	4.95	0.136 (35)	0.456 (26)	0.827
TabNAT†	42	1032 [996, 1074]	1025 [981, 1066]	0.187	8.76	0.159 (38)	0.400 (29)	0.910
Mean/mode	48	1000 [959, 1039]	998 [949, 1041]	0.195	10.42	0.165 (44)	0.443 (33)	0.904
TabPFN-3 (per cell)†	42	977 [943, 1013]	977 [943, 1020]	0.203	10.17	0.175 (39)	0.440 (30)	0.813
LimiX (per cell)	48	967 [925, 999]	970 [934, 1012]	0.175	6.52	0.140 (44)	0.499 (33)	0.816
ForestDiffusion†	42	955 [916, 995]	941 [890, 983]	0.188	11.88	0.173 (39)	0.462 (28)	0.924
MICE (t)†	47	840 [805, 871]	829 [795, 859]	0.193 (1 excl.)	9.66	0.156 (43)	0.525 (32)	0.817
GAIN (t)	48	838 [807, 870]	842 [804, 872]	0.189	9.54	0.154 (44)	0.522 (33)	0.815
GAIN	48	802 [768, 829]	802 [764, 832]	0.192	10.23	0.157 (44)	0.527 (33)	0.815
MICE†	46	666 [614, 715]	628 [566, 677]	0.229	13.07	0.194 (42)	0.550 (32)	0.810

Table 7: Class-conditional generation on the 36 classification tables: Elo with 95% bootstrap intervals and means over the tables each method completed (n). The Elo tasks are table × {utility, class-macro α-precision, class-macro β-recall, label-consistency gap}, where the gap is the absolute difference between the label consistency of the synthetic rows and that of held-out real rows. Label consistency is the fraction of synthetic rows assigned to the requested class by the scoring predictors trained on real rows, divided by the same fraction on held-out real rows (1 = as separable as real data; above 1 = classes separated more than in the real data); macro is the unnormalized macro-average over classes; agreement is the fraction of rows whose emitted label equals the requested class. TV and W_1 (median) are per-class distances, macro-averaged. (t) = the generation model with the hyperparameters tuned in its original paper. †: not completed on every table; a missing table is scored just below the worst method on that table in the Elo.

Method	n	Elo	Utility ↑	α-prec. ↑	β-rec. ↑	Consist. (rel.)	Consist. (macro)	Agreement	TV ↓	W_1 ↓
TabMosaic	36	1165 [1137, 1194]	0.926	0.624	0.349	0.983	0.675	1.00	0.064	0.096
TabPFN-3 (chained)†	35	1138 [1108, 1165]	0.915	0.606	0.359	0.985	0.675	1.00	0.086	0.100
TabNAT	36	1124 [1094, 1157]	0.881	0.675	0.381	0.972	0.671	1.00	0.048	0.117
TabEBM†	32	1062 [1025, 1097]	0.923	0.526	0.365	1.008	0.724	1.00	0.077	0.073
TabICLv2 (chained)	36	1025 [994, 1052]	0.911	0.449	0.267	0.941	0.665	1.00	0.084	0.367
ForestDiffusion†	27	1009 [979, 1055]	0.916	0.681	0.415	1.017	0.733	1.00	0.058	0.093
TabDDPM (t)†	33	997 [963, 1038]	0.826	0.570	0.345	0.909	0.669	1.00	0.124	0.546
GReaT†	30	973 [938, 1005]	0.849	0.656	0.376	0.953	0.656	1.00	0.168	0.271
TabPFGen†	31	962 [921, 999]	0.851	0.507	0.339	1.093	0.818	1.00	0.088	0.176
CTGAN (t)†	35	792 [751, 827]	0.597	0.415	0.206	0.794	0.466	0.70	0.172	0.292
LimiX (chained)†	32	752 [720, 786]	0.720	0.253	0.118	0.889	0.565	1.00	0.295	0.857
Real rows (held out)	36	–	–	–	–	1.000	0.709	–	–	–

Target-set scenarios, all methods. Table 8 extends Table 1 to the nine methods.

Table 8: Per-cell error on held-out rows relative to a mean/mode imputer (median over tables; below 1 = better than mean/mode), as the target set varies: the MCAR rate, the mechanism at 30%, and the number $|S|$ of whole hidden columns. 24 tables on which every method completed (the 15 smallest and 9 medium-sized ones; 23 for $|S|$=4). Best per row in bold.

		TabMosaic	TabNAT	ForestDiff.	DiffPuter	GAIN	MICE	TabPFN-3	TabICLv2	LimiX
MCAR rate	10%	**0.622**	0.777	1.000	0.901	1.001	1.157	0.912	0.729	0.872
	30%	**0.675**	0.860	1.082	0.966	1.028	1.233	1.098	0.812	0.916
	50%	**0.770**	0.932	1.113	0.955	1.141	1.237	1.212	0.921	0.966
	70%	**0.843**	0.993	1.182	1.005	1.290	1.296	1.328	0.974	1.016
30%	MAR	**0.628**	0.887	1.065	0.887	1.064	1.212	1.019	0.807	0.931
	MNAR	**0.704**	0.871	1.012	0.866	1.063	1.143	1.068	0.824	0.910
Columns	$\|S\|$=1	**0.523**	0.780	1.019	0.826	1.010	1.193	0.785	0.575	1.551
	$\|S\|$=2	**0.600**	0.895	1.053	0.822	1.001	1.158	0.992	0.776	1.702
	$\|S\|$=4	**0.767**	0.960	1.054	0.912	1.166	1.235	1.054	0.888	1.655
Tables with results (of 48)		48	48	36	42	48	48	28	48	48