DIFFPFN: FAST IN-CONTEXT TABULAR DIFFUSION VIA SYNTHETIC PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models for tabular data generation require training on each new dataset, making their use on new datasets computationally expensive. We introduce DiffPFN, an in-context diffusion model that generates tabular data without dataset-specific parameter updates. Pretrained on synthetic datasets drawn from a structural causal model prior, DiffPFN conditions its denoising process on observed training examples to generate samples with numerical and categorical features. Its transformer architecture jointly denoises all features, enabling a single pretrained model to operate across datasets with varying numbers and types of features. Experiments on eight datasets show competitive data fidelity, with a 39–$104\times$ speedup over diffusion baselines in total training and sampling time for new datasets. When fine-tuned, the pretrained model achieves higher downstream performance than a model trained from scratch at every budget up to 1,000 steps.

1 INTRODUCTION

Generating high fidelity synthetic tabular data has applications ranging from data augmentation and privacy protection to missing value imputation and counterfactual data generation (Yoon et al., 2018; Inan et al., 2023; Assefa et al., 2021). Generating synthetic data representative of a target distribution can help supplement or replace real data in machine learning workflows. Most methods proposed for this task are focused on fitting a model to each target dataset, and afterwards sampling new rows from the learned distribution (Kotelnikov et al., 2023; Kim et al., 2023; Lin et al., 2025).

Fitting a model for every new table requires dataset-specific training time. Prior-data fitted networks (Müller et al., 2022), such as TabPFN (Hollmann et al., 2025) and TabICL (Qu et al., 2025), use a pretrained Transformer trained on synthetic tasks from a tabular dataset prior to solve prediction tasks without dataset-specific training. Our goal is to extend this paradigm to joint tabular generation with a diffusion model.

We introduce DiffPFN, a conditional diffusion model for tabular data generation trained entirely on synthetic tasks. Pretraining consists of sampling tasks from a structural causal model prior (Qu et al., 2026), where the model is tasked to denoise corrupted rows based on clean rows sharing the same underlying structure. We use the categorical representation proposed by TabRep (Si et al., 2026), which employs a continuous representation for both numerical and categorical variables. The pretrained model is then able to generate new rows based on clean rows from an unseen table, without dataset-specific training (Figure 1).

We test DiffPFN on eight public tables against six dataset-specific generators and unsupervised generation with TabPFN, covering marginal and pairwise fidelity, detection, downstream utility, precision, coverage, a membership-inference diagnostic, and end-to-end new-task time. DiffPFN produces a complete synthetic table 39–$104\times$ faster than the diffusion baselines, with comparable marginal fidelity. Five hundred fine-tuning steps raise its column-pair rank to the best of the nine methods, at a cost comparable to training a dataset-specific model. Our code and pretrained weights are at `https://anonymous.4open.science/r/diffpfn-FF74/README.md`.

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.

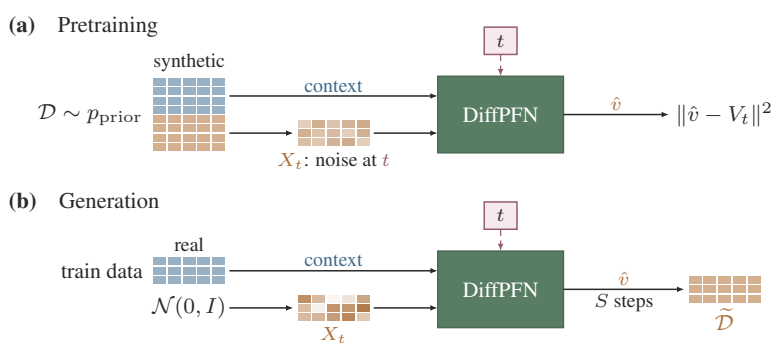


Figure 1: **Synthetic pretraining and generation on a new table.** (a) Each synthetic task is split into context rows and query rows. The queries are corrupted with Gaussian noise, and DiffPFN learns to predict their denoising velocities conditioned on the context. (b) For a new table, query rows are iteratively denoised and mapped back to the original numerical and categorical values.

2 Related Work

Dataset-specific tabular generators. Diffusion methods for tabular generation usually differ in how they represent and model numerical and categorical variables. TabDDPM (Kotelnikov et al., 2023) uses Gaussian diffusion for numerical columns and multinomial diffusion for one-hot categorical columns. TabSyn (Zhang et al., 2024) trains a variational autoencoder on the target table and applies diffusion in its latent space. TabDiff (Shi et al., 2025) uses a joint continuous-time process with learned feature-wise noise schedules. TabRep (Si et al., 2026) combines a numerical quantile transform with a circular categorical encoding, allowing both variable types to be modeled through continuous diffusion. These models train a generator on each target table. DiffPFN adopts TabRep's continuous representation but replaces dataset-specific model training with synthetic pretraining and in-context conditioning.

Prior-data fitted networks. Prior-data fitted networks are pretrained on tasks sampled from a prior and adapt through in-context learning. TabPFN (Hollmann et al., 2025) applies this approach to tabular prediction, while TabICL (Qu et al., 2025) extends it to larger tables. TabPFN's unsupervised mode also supports synthetic data generation, by sampling one column at a time conditioned on observed rows and previously generated columns. DiffPFN instead learns to jointly denoise all the variables in a row, so the number of denoiser evaluations is independent of table width, although the computational cost of each evaluation increases with the number of columns.

Cross-table and in-context generation. CTSyn (Lin et al., 2025) learns a shared latent space from real tables and reconstructs values with type-specific decoders; DiffICL (Han et al., 2026) pretrains a latent diffusion model on real datasets, conditions on context rows from the target table, and fits feature-wise decoders for each new dataset. DiffPFN is pretrained on synthetic tasks and returns generated rows to the original schema by quantile inversion and nearest-anchor categorical decoding, so no neural component is fitted to the target. Table 1 summarizes the comparison of DiffPFN with related approaches.

3 DiffPFN

3.1 Problem Formulation

Let $\mathcal{D} = \{\mathbf{z}^{(i)}\}_{i=1}^{N}$ be a table of rows drawn from an unknown distribution $p_{\mathcal{D}}$. Each row contains numerical and categorical features, $\mathbf{z}^{(i)} = (\mathbf{x}_{\text{num}}^{(i)}, \mathbf{x}_{\text{cat}}^{(i)})$, where $\mathbf{x}_{\text{num}}^{(i)} \in \mathbb{R}^{d_{\text{num}}}$ and $\mathbf{x}_{\text{cat}}^{(i)} \in \mathcal{C}_1 \times \cdots \times \mathcal{C}_{d_{\text{cat}}}$. Given context rows $\mathcal{D}_{\text{ctx}}$ from a previously unseen table, the objective is to approximate $p_{\mathcal{D}}$ and generate additional rows without updating model parameters.

Table 1: Positioning of DiffPFN against related approaches. *Fitted per new table* lists the components that receive gradient updates or are estimated from the target table before sampling.

Method	Pretraining data	Fitted per new table	Generation
Dataset-specific diffusion	None	Whole model	Per-table diffusion
CTSyn (Lin et al., 2025)	Real tables	Encoder, decoders	Schema-conditioned latent
DiffICL (Han et al., 2026)	Real tables	Decoders	Context-conditioned latent
TabPFN (Hollmann et al., 2025)	Synthetic	None	Autoregressive, per column
DiffPFN (ours)	Synthetic	Preprocessing only	Joint, all columns

During pretraining, each synthetic table is split into disjoint context and query sets. The context rows remain clean, and the query rows are corrupted by a diffusion process. The model learns to denoise query rows conditioned on the context, defining the conditional generative distribution

$$p_\theta \left(\mathcal{D}_{\mathrm{qry}} \mid \mathcal{D}_{\mathrm{ctx}} \right).$$

At inference, context rows come from the target training split, and query rows are initialized as Gaussian noise in the continuous representation. Iterative denoising produces synthetic rows conditioned on that context.

3.2 Synthetic pretraining prior

We train the model only on synthetic datasets, without using real-world data during pre-training. Training across datasets with different feature distributions and dependencies allows the model to learn to use the observed samples as context. We use the structural causal model prior of TabICLv2 (Qu et al., 2026) to generate synthetic datasets for training. Dataset size and the number of numerical and categorical features are sampled, together with a directed acyclic graph. Random functions define the relationships between parent and child nodes, including multilayer perceptrons, tree ensembles, Gaussian process functions, and linear and quadratic functions. Sampling random vectors at root nodes and applying these functions through the graph yields the node data. Features are then obtained from dimensions of randomly assigned nodes: a single dimension for numerical values, and multiple dimensions with discretization for categorical values. Dimensions not used for feature extraction remain unobserved, introducing noise. The graph structure and random functions provide different variable dependencies and levels of smoothness.

3.3 Unified continuous tabular representation

We represent numerical and categorical features in a unified continuous space following TabRep (Si et al., 2026). For numerical features, we apply a quantile transformation based on the target training rows to obtain a standard normal distribution. For a categorical feature with K distinct categories, we encode category $k \in \{0, \ldots, K-1\}$ as

$$\mathbf{c}_k = \left(\sqrt{2} \cos\left(\frac{2\pi k}{K} \right), \sqrt{2} \sin\left(\frac{2\pi k}{K} \right) \right).$$

With the factor $\sqrt{2}$, we give each categorical pair the expected squared norm of two standard-normal coordinates. Each categorical feature is represented in two dimensions, independent of its cardinality, and the total dimension is $d_{\mathrm{num}} + 2d_{\mathrm{cat}}$. This allows the same continuous diffusion process to model both feature types without output dimensions that are dependent on the number of categories. After generation, we apply the inverse quantile transformation to numerical features. Categorical samples are mapped to the nearest category in the circular representation.

3.4 Conditional diffusion objective

We use joint generation to process the features of each row together at each denoising step. This allows the number of denoising steps to remain independent of the number of columns, while attention across features allows the model to capture dependencies within each row. We train with a velocity

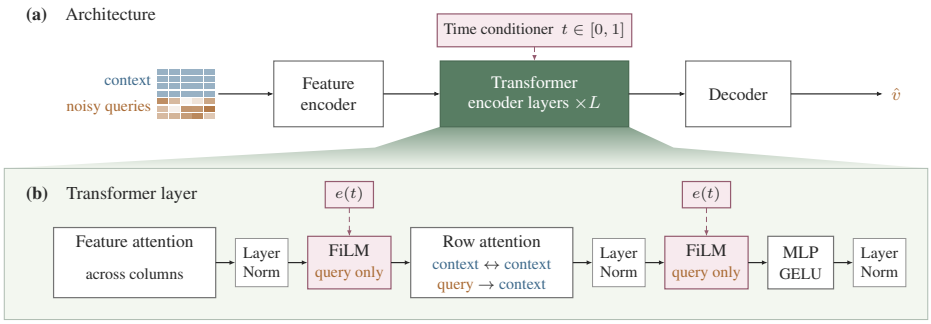


Figure 2: **DiffPFN architecture.** (a) A shared feature encoder embeds clean context and noisy query values. A stack of Transformer layers processes these embeddings, and a decoder predicts query velocities. (b) Each of the $L = 16$ layers applies attention across features and then across rows, with the time embedding $e(t)$ applied only to queries through FiLM (Perez et al., 2018).

parameterization (Salimans & Ho, 2022). For a clean row x_0 in the continuous representation and noise $\epsilon \sim \mathcal{N}(0, I)$, the forward process at timestep t gives

$$x_t = \alpha_t x_0 + \sigma_t \epsilon,$$

and the prediction target is

$$v_t = \alpha_t \epsilon - \sigma_t x_0.$$

Under the cosine schedule, v_t is the derivative of the noisy row with respect to the noise angle. The network predicts query velocities conditioned on the clean context $C = g(\mathcal{D}_{\text{ctx}})$, where g is the preprocessing map above. Stacking the query rows into X_t^{qry} and their targets into V_t^{qry}, we minimize the expected squared error

$$\mathcal{L} = \mathbb{E}_{\mathcal{D} \sim p_{\text{prior}}} \, \mathbb{E}_{(\mathcal{D}_{\text{ctx}}, \mathcal{D}_{\text{qry}})} \, \mathbb{E}_{t, \epsilon} \left[\| v_\theta(X_t^{\text{qry}}, t, C) - V_t^{\text{qry}} \|_2^2 \right].$$

3.5 Axial Transformer

DiffPFN uses a two-way attention mechanism following TabPFN (Hollmann et al., 2025) to capture relationships across both features and rows (Figure 2). Each cell in the continuous representation is encoded separately, allowing the same model to process tables with different numbers of rows and columns. Each layer first applies attention across features within each row, then across samples within each column. Attention across features allows the model to capture dependencies within a row, whereas attention across samples combines information from the observed data to guide generation. This structure allows the model to learn relationships between features in the context of each dataset.

During attention across rows, context rows attend only to context rows, and query rows attend only to context rows. Thus, query rows do not influence each other or the context representations. To represent the noise level, we apply feature-wise linear modulation (Perez et al., 2018) to the query representations using the diffusion timestep. The context representations are therefore independent of both the query samples and the timestep, allowing their reuse across denoising steps to avoid repeating computations on the observed data.

4 Experiments

We assess whether pretraining on synthetic data reduces the time required to generate data for new datasets while providing competitive generation quality. We compare DiffPFN with generative baselines on eight datasets, using data fidelity, downstream task performance, privacy, and total training and sampling time. We also examine how pretrained initialization, fine-tuning, context size, and denoising steps affect performance, and probe query representations to evaluate how the model captures relationships between features.

4.1 SETUP

Datasets. We evaluate DiffPFN on eight tabular datasets: Adult, Beijing, Car, Concrete Slump, Default, Ionosphere, Shoppers, and Wine Quality Red. The datasets contain fully numerical, fully categorical and mixed tables, with five classification tasks and three regression tasks. Training sets range from 92 to 37,581 rows. Adult uses the original training and test sets; for the remaining datasets, we use a 90/10 train/test split. Dataset details are provided in Appendix B.1 (Table 4).

Baselines. We compare DiffPFN and its fine-tuned model, DiffPFN-FT, with TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024), TabDiff (Shi et al., 2025), STaSy (Kim et al., 2023), CTGAN, TVAE (Xu et al., 2019), and TabPFN (Hollmann et al., 2025). TabPFN is used for unsupervised data generation. Baseline implementation details are presented in Appendix B.3. We did not include DiffICL (Han et al., 2026) and CTSyn (Lin et al., 2025) as baselines because neither had a publicly available implementation at the time of evaluation.

Metrics. We evaluate data fidelity, downstream task performance, and privacy of the synthetic data. Column-wise density estimation (CDE, column shapes) measures how well synthetic data captures individual feature distributions, while pairwise-column correlation (PCC, column-pair trends) measures relationships between features. The classifier two-sample test (C2ST, detection score) uses logistic regression to distinguish real from synthetic data. For C2ST, 1 corresponds to chance prediction and 0 to perfect prediction. Machine learning efficiency (MLE) evaluates a gradient-boosted tree model trained on synthetic data and tested on real test data, using AUROC for classification and RMSE for regression. We also report α-precision and β-coverage to evaluate fidelity and diversity, and membership inference attack (MIA) precision and recall as privacy diagnostics.

Implementation details. All methods use the same training split and generate the same number of synthetic rows as training rows. We average five sampling and evaluation runs for each dataset and method, with 360 runs in total. Each uses the same fitted model, so variation reflects sampling and evaluation randomness. DiffPFN uses 50 denoising steps, up to 1,000 context rows, and a generation batch size of 1,000. DiffPFN-FT uses the same checkpoint with 500 fine-tuning steps and a learning rate of 2×10^{-5}.

4.2 GENERATION ON UNSEEN TABLES

We compare the time required for model training and generating a new synthetic table, with DiffPFN and TabPFN excluding their pretraining times (Figure 3). Without training on the new dataset, DiffPFN achieves speedups of $39.2\times$, $43.6\times$, and $104.4\times$ over TabDDPM, TabSyn, and TabDiff, respectively, measured as the geometric mean of the time ratio across datasets. Figure 4 separates training and sampling time, showing the cost avoided by using a pretrained generator. Several baselines have faster sampling after training, thus the advantage of DiffPFN is the total time required to generate synthetic data for a new dataset.

Data fidelity. DiffPFN achieves competitive fidelity without further training, with the third-best average rank on C2ST. Its CDE score is higher than TabDiff on four of the eight datasets and TabSyn on five. Pairwise fidelity is less competitive. Its average PCC rank is 4.12, compared with 3.50 for TabDDPM and 3.25 for TabDiff. DiffPFN achieves an average α-precision rank of 3.50, matching TabDiff, while β-coverage is less competitive, with an average rank of 5.25 compared with 1.00 for TabDDPM.

Downstream task performance. DiffPFN has a worse average AUROC rank than the diffusion baselines, while outperforming CTGAN and TVAE. The difference is large on Car, where DiffPFN achieves 0.929 AUROC compared with 0.997 for TabDDPM. Regression performance varies across datasets: on Concrete Slump, DiffPFN has lower RMSE than TabDDPM and TabDiff, but on Beijing and Wine Quality Red its RMSE is higher. STaSy is excluded from the average classification ranks because its generated Car data does not contain all target classes.

Comparison with TabPFN. In order to compare our model with another pretrained baseline, we use unsupervised TabPFN. Figure 5 compares DiffPFN and TabPFN on each dataset. DiffPFN

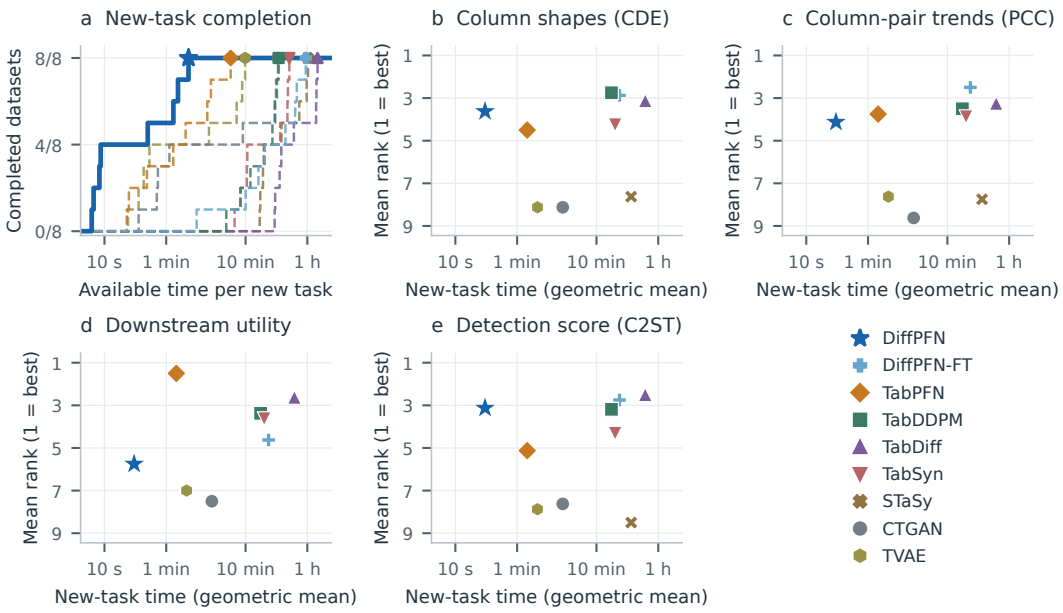


Figure 3: **New-task time and generation quality.** (a) Number of datasets for which each method completes training and generation within the indicated time budget. (b–e) Mean within-dataset ranks for column shapes (CDE), column-pair trends (PCC), downstream utility, and detection (C2ST), plotted against geometric-mean new-task time. Downstream utility ranks AUROC on the five classification tables and RMSE on the three regression tables within each dataset; STaSy is omitted from (d) because its Car runs produced no AUROC. Lower ranks and shorter times are better. New-task time includes target-table training, where required, and sampling-process time. Appendix C reports the remaining metrics and the per-dataset scores.

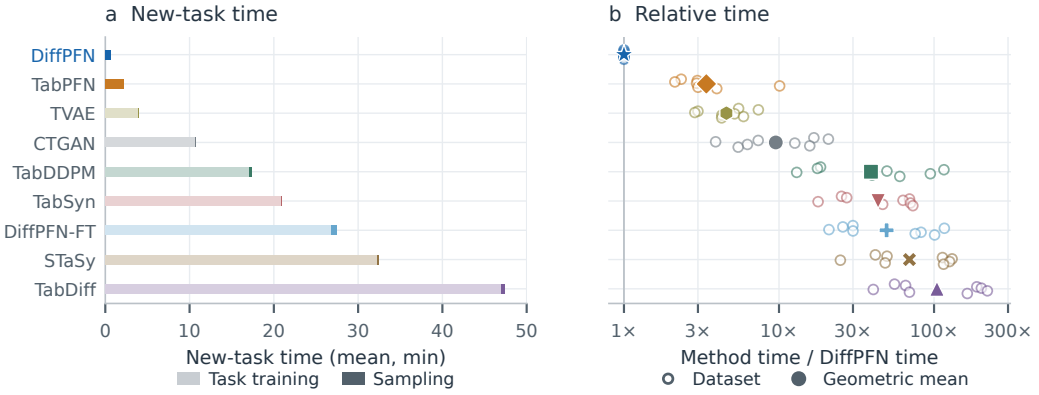


Figure 4: **Target-table training dominates new-task cost.** (a) Arithmetic-mean training and sampling wall time over the eight datasets. (b) Paired ratios to frozen DiffPFN, with geometric means. Frozen DiffPFN and TabPFN perform no target-table training. Each method was trained once per dataset, each open marker in (b) is a single measurement.

holds the higher detection score on all eight tables, and retains the higher column-shape score on five of eight. Column-pair trends divide four to four. TabPFN holds the better downstream MLE performance on all datasets. Regarding generation time, DiffPFN produces a complete synthetic table 2.1 to 10.1× faster.

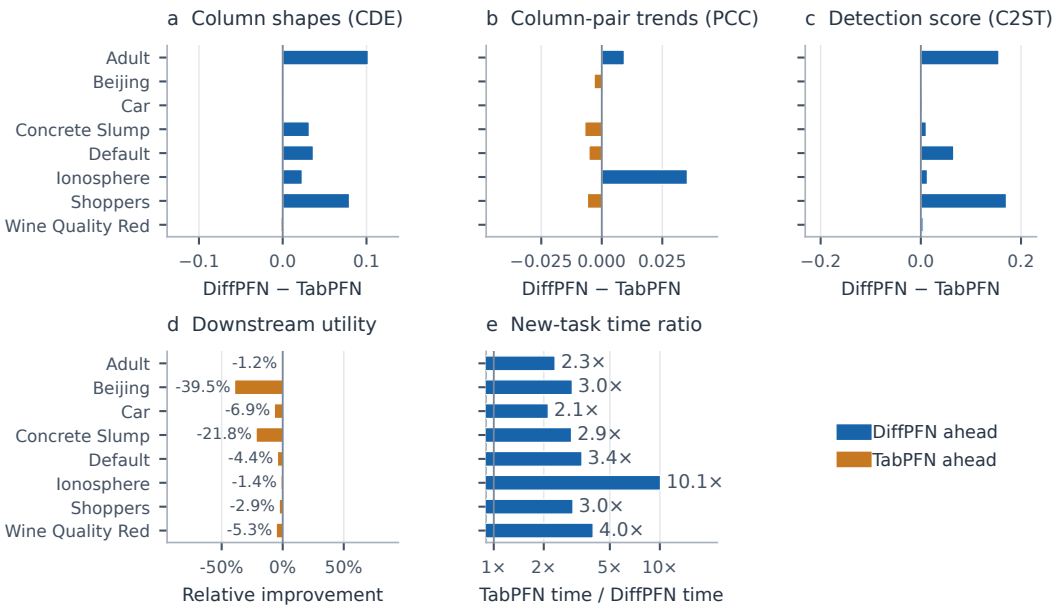


Figure 5: **Paired comparison with unsupervised TabPFN generation.** (a–c) Differences in column-shape, column-pair, and detection scores, computed as DiffPFN minus TabPFN. (d) Relative downstream improvement: the relative increase in AUROC for classification and reduction in RMSE for regression. Positive values favor DiffPFN. (e) TabPFN new-task time divided by DiffPFN new-task time; values above one indicate faster generation with DiffPFN.

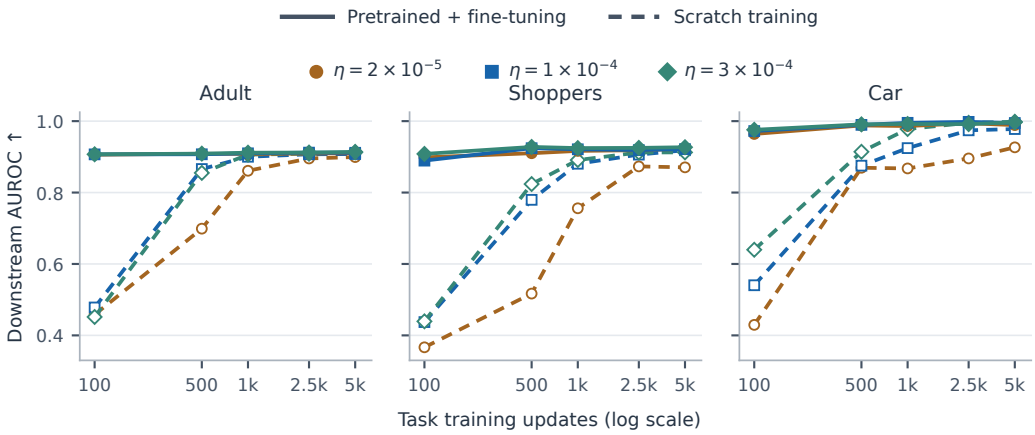


Figure 6: **Pretrained initialization accelerates adaptation to a new table.** Downstream AUROC against target-table updates, for the pretrained checkpoint with fine-tuning and the same architecture from random initialization. Color identifies the initial learning rate. Axes are shared and the update axis is logarithmic.

Fine-tuning. Allowing for fine-tuning on new tables provides several improvements. With 500 training steps, DiffPFN-FT achieves the best average PCC rank and improves β-coverage on all eight datasets. However, the downstream improvements are not consistent across datasets: classification performance improves on four datasets but is lower on Ionosphere. The total training and sampling time becomes comparable to TabDDPM and TabSyn.

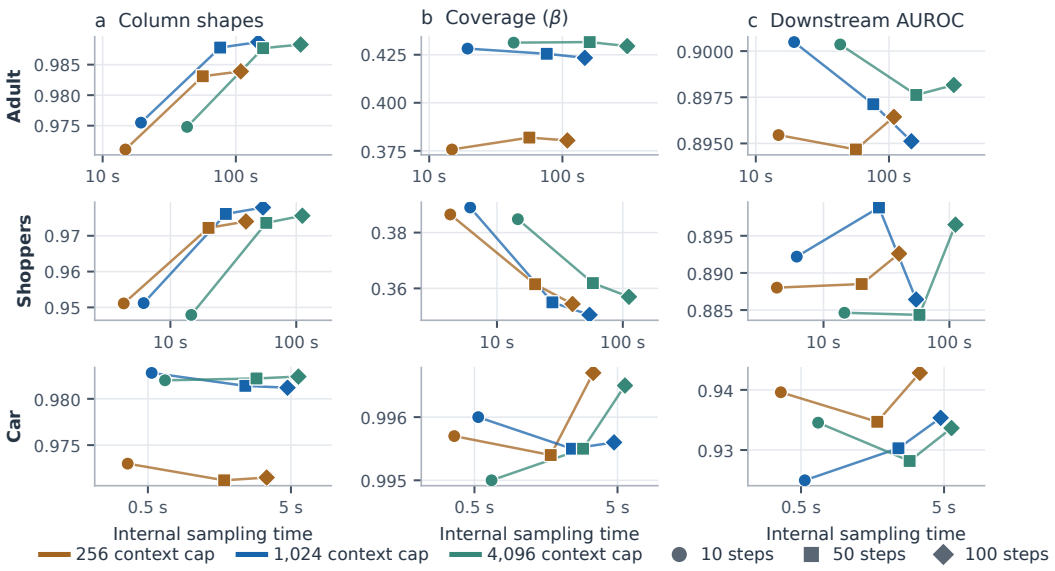


Figure 7: **Quality against measured sampling time.** All 27 recorded configurations: three datasets, three context caps, three step counts, one run each. The horizontal axis is time measured inside the generation routine, which excludes checkpoint loading and is not comparable with Figure 3. Lines join step counts at one context cap and are a reading aid. Car has 1,555 training rows, so its 4,096 setting uses all of them.

4.3 Model analysis

Pretrained initialization. We compare fine-tuning with training the same architecture from random initialization to assess whether pretraining reduces the target-table updates needed for useful generation. Both models use the same data, optimizer, training schedule, and evaluation protocol. On Adult, Shoppers, and Car, we evaluate three learning rates and five update budgets, from 100 to 5,000 steps (Figure 6).

Pretraining improves performance with limited training on the new dataset. Through 1,000 training steps, every pretrained model outperforms every model trained from random initialization on the same dataset, across the evaluated learning rates. The difference is smaller with further training, although the pretrained model achieves higher AUROC in 43 of the 45 comparisons with the same dataset, learning rate, and number of training steps.

Context size and denoising steps. We examine how inference computation affects generation quality on Adult, Shoppers, and Car, using context sizes of up to 256, 1,024, and 4,096 rows and 10, 50, and 100 steps. The number of generated samples is fixed for each dataset. For Car, the largest context setting uses all 1,555 training rows. Each configuration is evaluated once.

More denoising steps generally improve marginal fidelity and pairwise relationships on Adult and Shoppers, with smaller changes on Car. Figure 7 compares generation quality against sampling time. With up to 1,024 context rows, reducing the number of steps from 100 to 10 results in a 7.6–8.9$\times$ speedup in sampling. However, neither more steps nor larger context sizes consistently improve downstream classification performance or β-coverage.

Query representations. To evaluate how the model carries relationships between different columns, we probe intermediate query representations. We draw synthetic tasks from different function families and fit ridge probes on query embeddings to predict the row's true target. We also run a control diagnostic where the context's targets are independently shuffled before probe training and evaluation. We run the experiment on 8 function families, providing 256 rows as context and 32 as query, whose target entry is zero masked, with 56 tasks for probe training and 24 for evaluation. Figure 8 shows how target prediction increases across the Transformer layers, while performance with

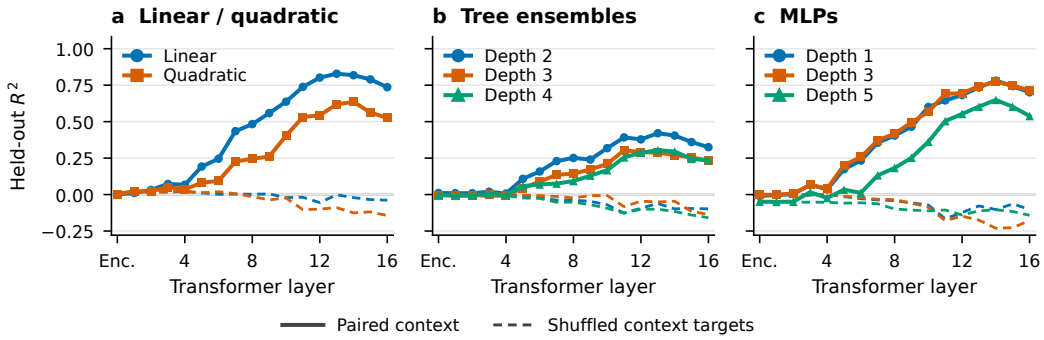


Figure 8: **Linear probes of context-conditioned query representations.** Held-out R^2 from ridge probes of pooled query embeddings at the encoder and successive transformer layers. Solid lines use paired context; dashed lines use shuffled context targets with separately fitted probes. Each family has 56 fitting and 24 held-out tasks; query targets are zero-masked and $t = 0$. One task seed; Appendix D.3 gives the protocol.

shuffled context targets remains close to zero. The highest R^2 is 0.83 for linear functions, whereas the shuffled control never exceeds 0.024 across the evaluated families and layers.

5 Conclusion

We introduce DiffPFN, a new architecture for tabular data generation that does not require dataset specific training. Pretraining on synthetic tasks from a structural causal model, allows the same model to generate both numeric and categorical data from an unseen table, without any parameter updates. On eight public tables, DiffPFN generates a new synthetic table 39–104$\times$ faster than the diffusion baselines, with competitive marginal fidelity and the third-best average rank on C2ST. Pairwise fidelity, β-coverage, and downstream task performance are less competitive. Fine-tuning further improves pairwise fidelity and coverage, although downstream improvements are not consistent across datasets. Our analysis shows that the model uses relationships in the observed context and that pretraining reduces the target-table updates needed for useful generation. These results extend the prior-data fitted network paradigm to joint tabular generation with a diffusion model.

Limitations. Our pretraining prior covers a restricted range of table sizes and categorical features. Future work could explore larger pretraining budgets and broader synthetic priors to improve generation across different target distributions. Extending the same pretrained model to missing value imputation and counterfactual data generation is another promising direction.

AI use statement

In this work, we used generative AI tools for polishing writing and for code development. We have not used generative AI tools for research ideation, design of the method, or as a component of the proposed method itself. We used generative AI tools for literature search and for drafting and formatting tables and figure captions. We have reviewed all AI-assisted work; AI-assisted code was reviewed and tested by the authors before use; and every citation was checked against the cited source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

The synthetic-data generation, pretraining, fine-tuning, sampling and evaluation code, the dataset preparation scripts and the pretrained checkpoint are released at `https://anonymous.4open.science/r/diffpfn-FF74/README.md`. Appendix B lists the dataset sources and splits,

the pretraining prior configuration, architecture and optimization hyperparameters, inference configuration, baseline configurations, evaluator implementations, and hardware used for the timing measurements.

ETHICS STATEMENT

Synthetic tabular data is often generated in order to share or augment data that cannot be released directly. We report membership-inference precision and recall for one attack at one operating point as a diagnostic. These numbers do not establish that generated tables protect the privacy of the individuals in the training data, and DiffPFN provides no formal privacy guarantee. Deploying it on personal data requires a separate privacy analysis. All eight evaluated datasets are public benchmark tables obtained from the UCI repository.

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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 290–306. PMLR, 2022.

Samuel A. Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E. Tillman, Prashant Reddy, and Manuela Veloso. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450375849. doi: 10.1145/3383455.3422554. URL https://doi.org/10.1145/3383455.3422554.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Xinyan Han, Yan Lu, Xiaoyu Lin, Yuanyuan Jiang, Yuanrui Wang, Xuanyue Li, Wenchao Zou, and Xingxuan Zhang. Breaking the quality-privacy tradeoff in tabular data generation via in-context learning. *arXiv preprint arXiv:2605.04911*, 2026.

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. doi: 10.1038/s41586-024-08328-6.

Muhammad Sakib Khan Inan, Sohrab Hossain, and Mohammed Nazim Uddin. Data augmentation guided breast cancer diagnosis and prognosis using an integrated deep-generative framework based on breast tumor's morphological information. *Informatics in Medicine Unlocked*, 37: 101171, 2023. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2023.101171. URL https://www.sciencedirect.com/science/article/pii/S2352914823000138.

Jayoung Kim, Chaejeong Lee, and Noseong Park. STaSy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2023.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17564–17579. PMLR, 2023.

Anton D. Lautrup, Tobias Hyrup, Arthur Zimek, and Peter Schneider-Kamp. SynthEval: A framework for detailed utility and privacy evaluation of tabular synthetic data. *Data Mining and Knowledge Discovery*, 39(1), 2025. doi: 10.1007/s10618-024-01081-4.

Xiaofeng Lin, Chenheng Xu, Matthew Yang, and Guang Cheng. CTSyn: A foundation model for cross tabular data generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SxsyLjIfWB.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

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.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Jacob Si, Zijing Ou, Mike Qu, Zhengrui Xiang, and Yingzhen Li. TabRep: Training tabular diffusion models with a simple and effective continuous representation. *Transactions on Machine Learning Research*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5689–5698. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/yoon18a.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.

A Method details

A.1 Pretraining prior

Synthetic tasks are drawn from the structural causal model prior of TabICLv2 (Qu et al., 2026), with the sampling parameters in Table 2. The prior varies the dependency graph and the conditional mechanisms, which include randomly initialized multilayer perceptrons and tree-based maps. Each sampled table receives a separate target column, which is appended to the feature block and treated

Table 2: Prior sampling parameters.

Parameter	Value
Prior type	`graph_scm`
Feature columns per table	2 to 20
Rows per table	200 to 1,000
Target classes	2 to 10
Categorical feature cardinality	2 to 200 (two-stage log-uniform)
Context fraction	$\mathcal{U}(0.5, 0.9)$ of the table's rows
Tasks per optimization batch	64
Tasks per shared-length group	8
Micro-batch size	8
Prior sampling device	CPU, generated online

as categorical, so a table with d feature columns is modeled as $d + 1$ columns. Tables within a micro-batch are padded to a common width and the padded coordinates are excluded from the loss.

Drawing 1,024 tables with the per-process batch of 8 used in training gives a mean of 10.2 feature columns and categorical feature cardinalities from 2 to 173, with median 3 and 12.2% of them above 10. The number of categorical feature columns per table ranges from 0 to 18, and 37% of the sampled tables contain none: the categorical schema is drawn once per per-process batch from the narrowest table in that batch. All eight evaluation tables have categorical column counts within this range.

A.2 Velocity parameterization

Under the cosine schedule the velocity target of Section 3 is a derivative of the noisy row. Writing $\alpha_t = \cos(\phi_t)$ and $\sigma_t = \sin(\phi_t)$ turns the forward process into a rotation in the plane spanned by x_0 and ϵ, so that $x_t = \cos(\phi_t)x_0 + \sin(\phi_t)\epsilon$. Differentiating with respect to the noise angle gives

$$\frac{\partial x_t}{\partial \phi_t} = -\sin(\phi_t)x_0 + \cos(\phi_t)\epsilon = v_t,$$

so v_t is the derivative of the noisy row with respect to the noise angle.

Since $\alpha_t^2 + \sigma_t^2 = 1$, a predicted velocity $\hat{v} = v_\theta(x_t, t, C)$ gives estimates of the clean row and the noise, $\hat{x}_0 = \alpha_t x_t - \sigma_t \hat{v}$ and $\hat{\epsilon} = \sigma_t x_t + \alpha_t \hat{v}$, without dividing by α_t, which vanishes at $t = 1$. Generation runs deterministic DDIM (Song et al., 2021) steps on the grid $t_k = k/S$, starting from $X_1 \sim \mathcal{N}(0, I)$ and setting $X_{t_{k-1}} = \alpha_{t_{k-1}}\hat{x}_0 + \sigma_{t_{k-1}}\hat{\epsilon}$. The last step has $\alpha_0 = 1$ and $\sigma_0 = 0$, so it returns $\hat{x}_0$.

A.3 Architecture and checkpoint

Every cell of the continuous representation is embedded by one shared linear map from a scalar to the embedding width, so the same parameters accept tables of any width and any length. Each of the 16 layers applies attention across the columns of a row, then attention across the rows of a column. Neither axis uses positional encoding. Row attention masks queries out of the context's key set and out of one another's, so context states are a function of the context rows alone and generated rows are conditionally independent given the context. Feature-wise linear modulation (Perez et al., 2018) applies the diffusion timestep to the query path twice per layer, after feature attention and after row attention, leaving the context path independent of t. A two-layer MLP decoder with GELU activation predicts one velocity per dense coordinate of each query row.

A.4 Algorithm

Table 3: Architecture and pretraining optimization.

Setting	Value
Transformer layers	16
Embedding width	256
Attention heads	4
Feed-forward width	1,024
Parameters	21,856,001
Gradient checkpointing	enabled
Diffusion schedule	cosine variance preserving, offset $s = 0.008$
Parameterization	velocity (Salimans & Ho, 2022)
Timestep sampling	$t \sim \mathcal{U}[0, 1]$, continuous
Optimizer	AdamW, betas $(0.9, 0.95)$, $\epsilon = 10^{-8}$
Learning rate	10^{-4}, cosine decay to 10^{-5}
Warmup	1,000 steps
Weight decay	0.01
Gradient-norm clipping	1.0
Precision	bfloat16 mixed
Steps	100,000
EMA decay	0.9995
Seeds	NumPy 0, PyTorch 0
Hardware	8× NVIDIA H100 80GB HBM3
Pretraining wall time	≈18 h

Algorithm 1 Pretraining on synthetic tasks (left) and generation conditioned on target-table rows (right). X denotes query rows in the continuous representation. The preprocessing map g uses statistics and category mappings estimated from data; g^{-1} denotes the numerical inverse transform and categorical decoding. The network factors as $v_\theta(X_t, t, C) = \mathrm{Dec}_\theta(X_t, t, \mathrm{Enc}_\theta(C))$, where Enc_θ returns the per-layer context states and Dec_θ is the query path (Section 3); Step applies one sampler update.

Pretraining

Require: prior p_{prior}, schedule $\{\alpha_t, \sigma_t\}$, model v_θ

while not converged **do**
  sample task $\mathcal{D} \sim p_{\mathrm{prior}}$, context size n_{ctx}
  split $\mathcal{D}$ into disjoint $\mathcal{D}_{\mathrm{ctx}}, \mathcal{D}_{\mathrm{qry}}$
  $C \leftarrow g(\mathcal{D}_{\mathrm{ctx}})$, $X_0 \leftarrow g(\mathcal{D}_{\mathrm{qry}})$
  $t \sim \mathcal{U}[0, 1]$, $\epsilon \sim \mathcal{N}(0, I)$
  $X_t \leftarrow \alpha_t X_0 + \sigma_t \epsilon$
  $V_t \leftarrow \alpha_t \epsilon - \sigma_t X_0$
  update θ using $\nabla_\theta \| v_\theta(X_t, t, C) - V_t \|_2^2$
end while

Generation

Require: train rows, v_θ, schedule $\{\alpha_t, \sigma_t\}$, $n_{\max}$, count N, steps S

fit g
draw $\mathcal{D}_{\mathrm{ctx}}$ of at most $n_{\max}$ rows
$C \leftarrow g(\mathcal{D}_{\mathrm{ctx}})$, $H \leftarrow \mathrm{Enc}_\theta(C)$
$X_1 \sim \mathcal{N}(0, I)$ with N rows; $t_k \leftarrow k/S$
for $k = S, \ldots, 1$ **do**
  $\hat{v} \leftarrow \mathrm{Dec}_\theta(X_{t_k}, t_k, H)$
  $X_{t_{k-1}} \leftarrow \mathrm{Step}(X_{t_k}, \hat{v}, t_k, t_{k-1})$
end for
return $g^{-1}(X_0)$

B Reproducibility

B.1 Datasets, sources, splits and preprocessing

All eight tables come from the UCI Machine Learning Repository. Adult uses the official `adult.data` and `adult.test` files. The remaining seven are split 90/10 into training and test partitions with seed 1234; for every table this split places each categorical level in the training partition. Beijing rows with missing values are dropped before splitting. Every generator is fitted on the training partition and asked for a synthetic table with the same number of rows; every evaluation uses the held-out test partition. The target column is generated jointly with the features and is never conditioned on.

Numerical columns are mapped through a quantile transform fitted on the training partition, with at most 1,000 quantiles and a standard-normal output distribution. Categorical columns are ordinal-

Table 4: Evaluated datasets. Counts exclude the target column. *Cardinality* is the largest number of distinct values in a categorical column, including the target.

Dataset	Train rows	Test rows	Num.	Cat.	Cardinality	Task
Adult	32,561	16,281	6	8	42	Binary
Beijing	37,581	4,176	6	5	31	Regression
Car	1,555	173	0	6	4	Multi-class
Concrete Slump	92	11	7	0	–	Regression
Default	27,000	3,000	14	9	11	Binary
Ionosphere	315	36	32	1	2	Binary
Shoppers	11,097	1,233	10	7	20	Binary
Wine Quality Red	1,439	160	10	0	–	Regression

encoded in the sorted order of the levels observed in the training partition, and each category k of a column of cardinality K is placed at $(\sqrt{2}\cos(2\pi k/K), \sqrt{2}\sin(2\pi k/K))$. Decoding inverts the quantile transform for numerical columns, maps integer-valued numerical columns to the nearest value observed in the training partition, and takes the nearest anchor in angle for categorical pairs. Neither map carries learned parameters, and both are fitted on the same rows the baselines use.

B.2 DiffPFN inference and fine-tuning

Generation draws a context of $\min(1{,}000, |\text{train}|)$ rows without replacement from the training partition, independently for each batch of generated rows. The per-layer context states are computed once per batch and reused across all 50 denoising steps. Sampling uses deterministic DDIM updates on the same cosine schedule as pretraining, a query batch size of 1,000, a context ensemble size of 1. The sampler also supports averaging velocities over several independently drawn context subsets.

Table 5: Inference settings.

Setting	Value
Context cap	1,000 rows
Context strategy	uniform without replacement, redrawn per batch
Context ensemble size	1
Denoising steps	50
Sampler	deterministic DDIM
Query batch size	1,000
Generated rows	one per training row

DiffPFN with target-table fine-tuning. DiffPFN-FT was initialized from the same pretrained EMA checkpoint and fine-tuned separately on each target dataset for 500 gradient updates. Fine-tuning used AdamW with learning rate 2×10^{-5}, weight decay 0.01, linear warm-up over the first 25 updates, and a minimum learning rate of 2×10^{-6}. Gradients were clipped to a norm of 1.0. Training used bfloat16 precision, an effective batch size of 32, microbatches of 16, EMA decay 0.99, and seed 0. Sampling used the dataset-specific EMA checkpoint and otherwise followed Table 5.

B.3 Baseline details

For each dataset, all dataset-specific models were trained on the real training partition and used to generate a synthetic dataset with the same number of rows. DiffPFN was evaluated both without parameter updates and after dataset-specific fine-tuning. Each method was trained once per dataset and sampled five times, with a new random seed for each sampling run.

The training and sampling settings used for each method are described below.

CTGAN. CTGAN was trained separately on each dataset for $1{,}000$ epochs with a batch size of 500. The categorical embedding dimension was $1{,}024$, and both the generator and discriminator

used hidden dimensions $(1{,}024, 2{,}048, 2{,}048, 1{,}024)$. The generator and discriminator were optimized with learning rates of 2×10^{-4}. Weight decay was 10^{-6} for the generator and zero for the discriminator.

TVAE. TVAE was trained separately on each dataset using an embedding dimension of $1{,}024$, encoder dimensions $(1{,}024, 2{,}048, 2{,}048, 1{,}024)$, decoder dimensions $(1{,}024, 2{,}048, 2{,}048, 1{,}024)$, batch size 500, and a reconstruction-loss factor of 2, for $1{,}000$ epochs. Optimization used Adam with weight decay 10^{-5} at the library's default learning rate of 10^{-3}.

STaSy. STaSy used a variance-exploding stochastic differential equation and an NCSN++-based tabular score network with hidden dimensions $(1{,}024, 2{,}048, 1{,}024, 1{,}024)$. Each dataset-specific model was trained for $10{,}000$ epochs using Adam with learning rate 2×10^{-3}, batch size $1{,}000$, 50 noise scales, $5{,}000$ warm-up steps, gradient clipping at 1.0, and EMA decay 0.9999.

TabDDPM. TabDDPM used an MLP denoising network with hidden dimensions $(1{,}024, 2{,}048, 2{,}048, 1{,}024)$ and no dropout, with separate Gaussian and multinomial diffusion processes for numerical and categorical variables. Numerical features were transformed using quantile normalization. Each dataset-specific model was trained for $100{,}000$ optimization steps using AdamW with learning rate 10^{-4}, batch size $4{,}096$ and $1{,}000$ diffusion timesteps. Weight decay was 5×10^{-4} on Adult, Car and Ionosphere, and 0 on Beijing, Concrete Slump, Default, Shoppers and Wine Quality Red.

TabDiff. TabDiff was trained separately on each dataset for $8{,}000$ optimization steps using AdamW with learning rate 10^{-3}, no weight decay, and a plateau schedule with factor 0.90 and patience 50, at batch size $4{,}096$. Its denoising network used two layers, token dimension 4, one attention head, feed-forward expansion factor 32, and timestep dimension $1{,}024$. Model parameters were tracked using an exponential moving average with decay 0.997. The implementation used 50 diffusion timesteps and the stochastic second-order sampler for generation.

TabPFN generation. The TabPFN generator required no gradient-based dataset-specific training. Its unsupervised density estimator was fitted in context when sampling from each target table. Generation used three estimators, three feature permutations, and a sampling temperature of 1.0.

TabSyn. TabSyn was trained in two stages. First, its variational autoencoder was trained for $4{,}000$ epochs with batch size $4{,}096$, using Adam at learning rate 10^{-3} with no weight decay and a plateau schedule with factor 0.95 and patience 10. The VAE used a token dimension of 4, two transformer layers, one attention head, and a feed-forward expansion factor of 32, with the adaptive KL-weight schedule $\beta_{\max} = 10^{-2}$, $\beta_{\min} = 10^{-5}$, and $\lambda = 0.7$. The latent diffusion model used a denoising MLP with hidden dimension $1{,}024$ and was trained using Adam with learning rate 10^{-3} and batch size $4{,}096$. Training ran for at most $10{,}001$ epochs, with a plateau learning-rate schedule and early stopping after 500 epochs without improvement. Sampling used 50 reverse diffusion steps.

B.4 Evaluator definitions

We use the metrics reported by TabRep (Si et al., 2026) and the evaluation code of TabSyn (Zhang et al., 2024). Downstream utility is measured by machine learning efficiency (MLE). Low-order statistics are column-wise density estimation (CDE) and pairwise column correlation (PCC). High-order statistics are α-precision and β-coverage (Alaa et al., 2022). We report a classifier two-sample test (C2ST) and the precision and recall for membership inference attack (MIA) (Shokri et al., 2017).

Machine learning efficiency. An XGBoost model (Chen & Guestrin, 2016) is trained on the synthetic table and evaluated on the held-out real test partition. Hyperparameters are selected on a validation split drawn from the synthetic table, so no real rows enter model selection. Classification tasks report the area under the ROC curve,

$$\mathrm{AUROC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR})\, d\,\mathrm{FPR},$$

using the one-versus-rest macro average on the multiclass Car table. Higher values are better. Regression tasks report

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

where lower values are better. Following the evaluation protocol of the dataset-specific baselines, regression targets are mapped to $\log(\mathrm{clip}(y, 1, 20{,}000))$ in the training, validation and test sets. The lower clip raises exact zeros to one.

Low-order statistics. For a numerical column, CDE uses the Kolmogorov–Smirnov statistic between the empirical distribution functions F_1 and F_2 of the real and synthetic values,

$$\mathrm{KST} = \sup_x |F_1(x) - F_2(x)|, \qquad F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(-\infty,x]}(X_i).$$

For a categorical column, it uses the total variation distance between the real and synthetic category frequencies P_1 and P_2,

$$\mathrm{TVD} = \frac{1}{2}\sum_{x\in\mathcal{X}} |P_1(x) - P_2(x)|.$$

For a pair of numerical columns, PCC compares Pearson correlations $\rho(x, y) = \mathrm{cov}(x, y)/(\sigma_x\sigma_y)$ through

$$\text{Pearson score} = \frac{1}{2}\,\mathbb{E}_{x,y}\,|\rho_1(x, y) - \rho_2(x, y)|.$$

For a pair involving a categorical column, it compares the joint frequency tables P_1 and P_2 of the two columns,

$$\text{Contingency score} = \frac{1}{2}\sum_{\alpha\in A}\sum_{\beta\in B}\left|P_{1,(\alpha,\beta)} - P_{2,(\alpha,\beta)}\right|,$$

where A and B are the levels of the two columns. Numerical columns are binned before the contingency score is computed for mixed pairs. Each distance is converted to a similarity in $[0, 1]$ as one minus the distance, and CDE and PCC average these similarities over columns and column pairs, so higher values are better. We use the SDMetrics implementation.[1]

High-order statistics. α-precision and β-coverage (Alaa et al., 2022) extend precision and recall to generative models. α-precision assumes that an α fraction of the real rows is typical of the data distribution and the rest are outliers, and measures the fraction of synthetic rows that fall in the region of those typical real rows. It measures fidelity. β-coverage applies the same construction to a β fraction of the synthetic rows and measures the fraction of the real distribution they cover. It measures diversity. Both are computed by SynthCity[2] directly on the one-hot encoded rows, without a learned embedding, and higher values are better.

Classifier two-sample test. C2ST measures how well a classifier separates real from synthetic rows. SDMetrics fits a logistic regression under cross-validation and returns one minus the mean of $2\max(\mathrm{AUROC}, 0.5) - 1$. A score of 1 corresponds to a classifier at chance and 0 to perfect separation, so higher values are better.

Membership inference. A membership inference attack tries to determine whether a given row was part of the generator's training data (Shokri et al., 2017). We use the attack implemented in SynthEval (Lautrup et al., 2025), which trains a classifier to separate synthetic rows from held-out real rows and evaluates it on a balanced mixture of training and held-out rows. We report its precision, the fraction of rows predicted as members that are members, and its recall, the fraction of members that are identified. On a balanced mixture, random membership predictions have expected precision 0.5, while recall depends on the fraction of rows predicted as members. Precision and recall should therefore be interpreted jointly.

[1] https://github.com/sdv-dev/SDMetrics

[2] https://github.com/vanderschaarlab/synthcity

Table 6: Mean within-dataset rank, lower is better. Each column uses a dataset set that is fixed across the methods it includes; the set and method count are given in the header. AUROC omits STaSy on every dataset because its five Car runs produced no downstream score. RMSE is measured on the transformed targets described in Appendix B.4.

	Fidelity (8 ds, 9 m)			Utility		Quality (8 ds, 9 m)	
Method	CDE	PCC	C2ST	AUROC (5 ds, 8 m)	RMSE (3 ds, 9 m)	α	β
CTGAN	8.12	8.62	7.62	7.60	8.33	7.94	8.38
TVAE	8.12	7.62	7.88	6.60	8.67	8.00	8.00
STaSy	7.62	7.75	8.50	–	6.67	7.00	6.75
TabPFN	4.50	3.75	5.12	**1.40**	**1.67**	4.00	3.50
TabDDPM	**2.75**	3.50	3.19	3.80	2.67	3.88	**1.00**
TabDiff	3.12	3.25	**2.50**	2.00	3.67	3.50	3.88
TabSyn	4.25	3.88	4.31	3.60	3.67	**3.38**	4.62
DiffPFN	3.62	4.12	3.12	6.20	5.33	3.50	5.25
DiffPFN-FT	2.88	**2.50**	2.75	4.80	4.33	3.81	3.62

C COMPLETE RESULTS

This section reports every recorded benchmark quantity: mean ranks across all seven metrics (Table 6), a per-column view of three generated tables against their training splits (Figure 9), the metric panels omitted from Figure 3 (Figure 10), and per-dataset means with sample standard deviations for all nine methods (Tables 7 and 8).

Repeated runs. We executed five sampling and evaluation runs per dataset–method pair and one training run. The dispersion in Tables 7 and 8 therefore reflects run-to-run variation in sampling and evaluation and no entry is a significance test.

Generated tables against their training splits. Figure 9 shows one stored generation instance per dataset for Car, Shoppers and Adult, sampled at the operating point of Table 5. Across the seven categorical columns shown, the total variation distance between real and synthetic category shares is below 0.04, and no synthetic row contains a category absent from the training split. Pairs with a strong real association keep most of it. On Adult, `education.num` with `education` goes from 1.000 to 0.984, and on Shoppers, `BounceRates` with `ExitRates` goes from 0.912 to 0.892. The mean absolute association gap over categorical pairs is 2.9 times the gap over numerical pairs on Shoppers (0.120 against 0.041) and 2.1 times on Adult (0.046 against 0.022). The largest categorical-pair gap on each table is `OperatingSystems` with `Browser` on Shoppers, from 0.602 to 0.074, and `race` with `native.country` on Adult, from 0.409 to 0.045.



Figure 9: **Generated tables against their real training splits, for three datasets. (a)** Marginals for one numerical column, one categorical column and the target of each table; grey is the real training split, blue the frozen DiffPFN sample. **(b)** Pairwise column association in the real table, in the synthetic table, and their absolute difference. The sign of the numerical correlations is dropped. Columns are sorted numerical before categorical with the target last. The diagonal is masked.

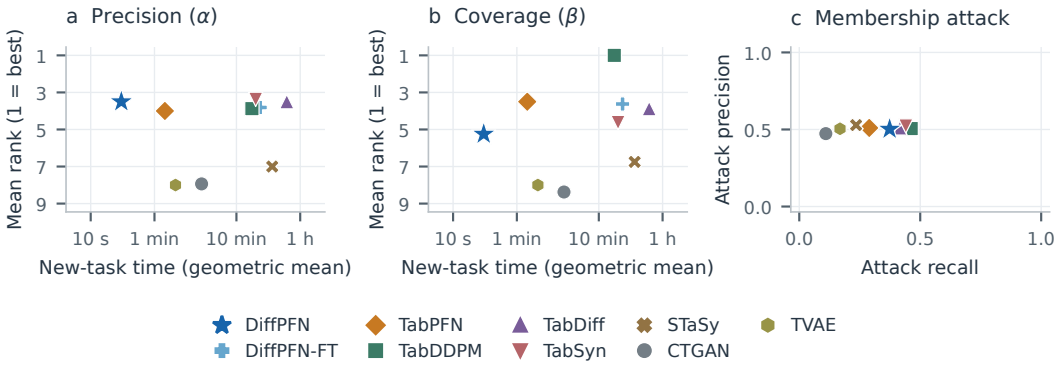


Figure 10: **Metrics omitted from Figure 3.** (a) α-precision and (b) β-coverage mean within-dataset ranks against geometric-mean new-task time, on all eight datasets and nine methods; lower rank is better. (c) Mean membership-attack precision against mean recall, averaged over the five datasets on which every method's attack predicted at least one member (Car, Default, Ionosphere, Shoppers and Wine Quality Red); when the attack predicts no members its precision is undefined. Ranks use mean scores within each dataset, with average ranks for ties.

Table 7: **Per-dataset results, part 1 of 2: Adult, Beijing, Car and Concrete Slump.** Mean ± sample standard deviation over five sampling and evaluation runs. CDE, PCC, C2ST, α, β and the two membership-inference columns are scaled by 100; AUROC and RMSE are on their natural scales. *New-task* is recorded target-table training time plus sampling-process wall time; frozen DiffPFN and TabPFN do no target-table training. Best per column in bold; the membership-inference columns have no preferred direction and are not ranked. Best per column in bold; the membership-inference columns have no preferred direction and are not ranked. MIA-P averages only the runs in which the attack predicted at least one member; when it predicts none, precision is undefined, and a value without ± comes from a single run. −− marks a metric that does not apply or was not produced, including MIA-P when no run defines it.

Adult

Method	AUROC ↑	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.872±0.002	81.40±0.04	80.57±0.20	52.22±0.30	81.14±0.45	13.81±0.08	20.00	0.00±0.00	1417.2±0.3
TVAE	0.879±0.003	75.62±0.05	70.77±1.41	26.50±0.51	54.61±0.27	13.52±0.10	–	0.00±0.00	463.4±0.1
STaSy	0.904±0.001	85.69±0.09	80.44±0.16	30.65±0.71	83.62±0.33	33.66±0.31	50.20±0.31	19.96±0.21	3530.4±0.4
TabDDPM	0.909±0.004	98.43±0.07	97.46±0.13	90.23±0.49	91.29±0.42	**62.65**±0.22	50.93±0.25	55.58±0.60	1564.6±0.8
TabDiff	**0.911**±0.001	**99.30**±0.04	**98.58**±0.07	**99.28**±0.27	98.51±0.25	52.62±0.32	51.17±0.15	51.44±1.32	4695.2±1.4
TabSyn	0.910±0.002	98.68±0.04	97.73±0.05	96.00±0.41	**99.42**±0.08	47.95±0.15	50.72±0.18	49.06±1.13	2136.5±0.4
TabPFN	0.909±0.001	88.58±0.06	95.31±0.11	79.97±0.53	91.26±4.18	51.03±1.03	50.61±5.15	0.29±0.02	197.6±0.5
DiffPFN	0.899±0.002	98.80±0.11	96.24±0.13	95.56±0.72	98.74±0.21	42.21±0.21	50.29±0.30	34.67±1.29	**84.4**±0.8
DiffPFN-FT	0.907±0.002	99.21±0.07	97.95±0.11	98.24±0.61	99.33±0.06	46.16±0.16	50.52±0.28	50.38±1.27	2530.6±0.2

Beijing

Method	RMSE ↓	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.924±0.013	80.90±0.02	71.00±0.08	74.44±0.32	94.16±0.10	39.24±0.21	–	0.00±0.00	1538.7±0.3
TVAE	1.055±0.010	73.85±0.05	84.25±0.57	41.99±1.64	81.72±0.25	2.88±0.07	–	0.00±0.00	543.9±0.4
STaSy	0.636±0.003	90.76±0.13	90.85±0.37	67.17±0.19	92.20±0.31	47.25±0.23	50.00±0.88	32.77±0.64	3674.4±0.3
TabDDPM	**0.470**±0.004	**99.10**±0.02	97.17±0.81	97.82±0.23	97.52±0.31	**73.05**±0.32	50.83±0.57	48.17±0.87	1307.6±0.6
TabDiff	0.554±0.002	97.82±0.02	96.84±0.06	**97.97**±0.18	98.15±0.28	59.59±0.27	51.01±0.68	19.75±0.92	4842.9±0.4
TabSyn	0.633±0.003	96.34±0.12	95.10±0.09	88.97±0.53	98.45±0.17	51.05±0.31	50.53±0.44	41.39±0.95	2019.5±0.3
TabPFN	0.473±0.006	98.12±0.08	**97.89**±0.41	97.32±1.39	**99.09**±0.28	72.32±0.66	50.56±13.78	0.23±0.11	219.6±1.6
DiffPFN	0.660±0.003	98.01±0.09	97.57±0.40	97.46±0.36	99.08±0.03	49.15±0.12	50.09±0.81	16.67±0.89	**73.9**±0.5
DiffPFN-FT	0.630±0.001	98.07±0.05	97.89±0.42	97.89±0.25	98.85±0.30	53.22±0.12	50.82±1.09	19.57±0.92	1906.8±0.2

Car

Method	AUROC ↑	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.887±0.021	79.37±0.19	70.13±0.48	58.13±0.51	72.99±0.66	92.50±1.09	48.55±2.79	27.73±1.98	65.8±0.5
TVAE	0.889±0.017	73.43±0.55	79.79±0.95	34.39±2.24	84.07±0.62	60.58±0.98	52.01±2.65	18.46±1.31	26.9±0.2
STaSy	–	74.35±0.15	59.71±0.66	31.08±0.80	80.89±1.20	80.66±1.01	54.45±2.65	22.18±2.75	1015.4±0.3
TabDDPM	0.997±0.001	98.34±0.18	97.37±0.40	99.63±0.41	97.43±0.85	**99.62**±0.04	49.72±2.59	48.37±3.59	1038.0±0.5
TabDiff	0.991±0.003	98.34±0.40	97.05±0.56	99.56±0.98	**98.48**±0.16	99.61±0.09	49.91±1.93	48.00±2.37	1691.0±0.3
TabSyn	0.974±0.004	98.05±0.15	96.66±0.54	99.08±1.14	97.86±1.12	99.61±0.08	53.75±3.30	55.18±3.69	622.4±0.3
TabPFN	**0.998**±0.001	**98.55**±0.24	97.72±0.54	99.52±0.44	97.90±1.20	99.54±0.10	51.00±1.74	50.73±2.46	19.1±0.4
DiffPFN	0.929±0.019	98.54±0.32	97.74±0.32	**99.85**±0.31	97.75±0.82	99.54±0.12	50.67±2.27	51.55±3.87	**9.0**±0.5
DiffPFN-FT	0.983±0.004	98.23±0.26	**97.75**±0.49	98.90±1.01	97.68±0.54	99.58±0.05	49.66±2.34	52.73±4.08	1044.5±0.3

Concrete Slump

Method	RMSE ↓	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.248±0.038	76.90±0.90	83.94±1.99	45.39±5.46	82.34±2.48	16.74±2.05	–	0.00±0.00	27.0±0.5
TVAE	0.268±0.014	81.88±1.75	86.31±2.92	90.18±5.67	76.43±0.79	39.16±5.49	–	0.00±0.00	19.8±0.3
STaSy	0.191±0.035	57.23±1.83	88.49±1.59	9.19±3.21	60.98±8.01	25.55±5.06	–	0.00±0.00	903.2±0.3
TabDDPM	0.140±0.021	90.73±1.34	96.19±0.57	98.63±1.61	93.76±3.37	**75.89**±5.33	–	0.00±0.00	343.2±0.2
TabDiff	0.166±0.050	89.89±1.20	97.18±0.94	98.25±3.36	91.94±1.42	39.96±3.62	–	0.00±0.00	1394.9±0.3
TabSyn	**0.098**±0.026	89.18±0.43	**97.37**±0.86	98.19±3.34	90.66±2.68	72.80±6.56	–	0.00±0.00	435.8±0.4
TabPFN	0.111±0.044	88.42±1.12	96.54±0.98	98.90±1.85	**95.12**±1.68	58.42±6.99	–	0.00±0.00	20.3±0.7
DiffPFN	0.136±0.039	**91.60**±1.17	95.84±1.17	**100.00**±0.00	92.95±1.62	59.98±4.55	–	0.00±0.00	**6.9**±0.5
DiffPFN-FT	0.150±0.036	90.60±1.28	96.28±1.78	99.31±0.97	82.34±4.65	65.12±3.88	–	0.00±0.00	145.0±0.3

Table 8: **Per-dataset results, part 2 of 2: Default, Ionosphere, Shoppers and Wine Quality Red.** Columns, scaling and conventions as in Table 7.

Default

Method	AUROC ↑	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.745±0.002	82.76±0.04	71.25±0.27	43.29±0.56	66.42±0.26	11.33±0.18	53.71±9.56	0.50±0.09	1448.9±0.4
TVAE	0.741±0.005	92.36±0.04	95.13±0.10	69.42±0.46	93.33±0.19	30.61±0.17	51.04±4.19	1.61±0.32	592.7±0.5
STaSy	0.754±0.004	93.68±0.11	93.58±0.20	61.99±0.20	95.12±0.15	39.94±0.33	51.44±0.54	32.11±0.31	2851.2±0.4
TabDDPM	0.754±0.006	98.19±0.08	97.91±0.09	93.34±0.66	96.81±0.31	**53.19**±0.19	51.00±0.44	44.93±0.77	1490.2±0.7
TabDiff	0.764±0.003	**98.72**±0.08	**98.66**±0.07	**97.01**±0.79	**98.70**±0.23	51.35±0.19	50.72±0.61	46.94±0.49	4647.4±1.0
TabSyn	0.761±0.002	98.05±0.05	98.09±0.11	88.94±0.53	98.63±0.23	45.80±0.26	51.01±0.69	39.17±0.68	2047.1±0.3
TabPFN	**0.765**±0.003	93.51±0.14	96.35±0.25	88.72±0.45	92.91±0.56	49.41±0.35	50.24±4.12	1.30±0.16	389.3±1.1
DiffPFN	0.731±0.004	97.17±0.16	95.82±0.07	95.30±0.81	94.31±0.38	38.51±0.32	50.36±0.95	22.51±0.51	**114.6**±0.4
DiffPFN-FT	0.757±0.002	98.48±0.05	98.08±0.13	96.83±0.45	97.67±0.40	46.47±0.21	50.22±0.68	46.53±1.40	3438.3±0.3

Ionosphere

Method	AUROC ↑	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.595±0.253	68.53±0.61	66.24±0.43	5.44±1.18	34.40±0.15	0.56±0.54	43.73±15.70	12.45±5.58	45.6±0.2
TVAE	0.943±0.018	76.91±0.58	73.08±0.84	40.91±1.59	73.21±0.86	12.24±1.50	53.71±8.50	39.11±9.38	31.2±0.6
STaSy	0.775±0.054	70.95±0.66	76.81±0.61	3.25±0.80	84.71±2.25	4.94±1.01	56.86±13.26	15.11±2.44	919.0±0.4
TabDDPM	0.952±0.021	92.84±0.54	90.89±0.82	**99.98**±0.05	**93.25**±2.39	**81.10**±1.50	49.49±4.40	45.78±4.61	690.6±0.3
TabDiff	0.972±0.012	87.57±0.35	78.32±0.58	99.61±0.58	62.13±2.12	4.40±0.25	53.24±6.36	39.11±8.97	1611.0±0.4
TabSyn	0.966±0.010	86.24±0.74	78.11±0.58	76.81±7.77	62.22±1.29	3.93±0.39	52.74±4.18	36.00±5.53	513.8±0.3
TabPFN	**0.973**±0.010	89.99±0.60	86.53±0.88	97.21±3.01	91.64±2.44	55.49±2.84	55.65±4.40	46.22±8.07	73.6±2.9
DiffPFN	0.959±0.016	92.32±0.92	90.07±0.72	98.53±2.94	84.25±3.16	47.37±2.21	49.31±7.83	44.45±8.61	**7.3**±0.2
DiffPFN-FT	0.929±0.030	**94.25**±0.43	**92.33**±0.55	99.91±0.21	90.93±2.22	55.55±1.39	53.36±8.48	54.67±9.51	601.5±0.3

Shoppers

Method	AUROC ↑	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.844±0.012	74.55±0.09	81.50±0.09	57.59±0.36	72.04±0.12	19.56±0.18	39.60±23.04	0.32±0.18	554.1±0.4
TVAE	0.901±0.005	74.98±0.05	84.85±0.24	22.94±0.69	39.54±0.33	16.79±0.21	40.67±15.50	0.32±0.05	207.2±0.4
STaSy	0.903±0.007	83.59±0.15	84.26±0.17	37.16±0.45	73.71±0.70	29.89±0.26	48.04±1.86	20.09±0.74	1698.5±6.1
TabDDPM	0.918±0.006	97.59±0.04	97.15±0.33	87.91±0.69	90.26±0.49	**72.22**±0.20	50.87±0.83	45.24±1.10	1398.2±0.7
TabDiff	0.922±0.005	98.05±0.08	98.06±0.17	**98.21**±0.85	98.78±0.51	51.01±0.58	51.22±1.79	37.06±1.72	2436.0±1.1
TabSyn	0.911±0.005	**98.32**±0.05	98.34±0.33	97.36±0.65	**99.23**±0.19	47.61±0.41	52.09±0.55	46.43±0.69	1636.7±0.5
TabPFN	**0.924**±0.004	89.37±0.60	94.81±0.33	74.35±1.33	89.24±0.58	48.51±0.48	46.07±4.74	4.47±0.29	105.3±1.2
DiffPFN	0.897±0.009	97.34±0.09	94.22±0.37	91.44±0.83	97.43±0.34	36.47±0.83	50.87±1.24	31.36±1.61	**35.1**±0.7
DiffPFN-FT	0.914±0.009	98.24±0.05	**98.40**±0.09	95.86±0.71	98.09±0.59	46.57±0.60	50.46±1.32	35.78±0.72	2653.7±0.2

Wine Quality Red

Method	RMSE ↓	CDE ↑	PCC ↑	C2ST ↑	α ↑	β ↑	MIA-P	MIA-R	New-task (s) ↓
CTGAN	0.160±0.003	82.62±0.37	78.61±0.58	36.41±1.64	92.67±2.06	10.33±0.34	50.98±6.19	13.90±2.90	47.1±0.3
TVAE	0.144±0.007	81.15±0.29	94.21±0.60	42.27±0.96	85.17±0.74	18.84±0.71	54.86±1.91	24.70±2.14	36.9±0.6
STaSy	0.137±0.009	80.88±0.42	93.90±0.39	24.36±0.25	87.68±1.23	33.60±0.95	53.43±2.12	28.00±1.66	994.8±0.4
TabDDPM	0.130±0.005	**96.80**±0.39	**99.21**±0.25	**99.75**±0.34	98.11±1.05	**56.84**±0.78	51.86±1.56	47.00±2.72	519.4±0.4
TabDiff	0.129±0.005	95.58±0.29	99.14±0.25	99.49±0.99	96.83±0.98	28.45±0.92	50.07±4.77	37.10±4.71	1415.9±0.4
TabSyn	0.131±0.004	95.92±0.18	99.03±0.25	**99.75**±0.55	96.72±1.45	31.46±0.61	51.63±1.78	43.40±3.44	630.6±0.2
TabPFN	**0.128**±0.004	95.56±0.09	99.14±0.48	99.24±0.73	94.58±1.87	39.68±1.45	52.10±2.26	41.90±2.61	34.3±0.6
DiffPFN	0.135±0.003	95.35±0.14	99.20±0.12	99.70±0.66	**98.37**±0.34	35.36±1.60	49.99±3.11	36.90±3.31	**8.6**±0.5
DiffPFN-FT	0.130±0.005	95.38±0.19	98.94±0.28	99.73±0.48	95.04±1.29	41.05±0.80	49.96±1.77	36.20±3.47	871.0±0.1

D MODEL ANALYSIS

D.1 PRETRAINING VERSUS SCRATCH

Table 9 lists every one of the 45 matched pretrained/scratch settings behind Figure 6: three datasets, three initial learning rates and five update budgets, in both initialization modes.

D.2 INFERENCE-TIME CONTROLS

Table 10 gives every recorded configuration of the context/steps sweep: three datasets, three requested context caps and three denoising-step counts, one run each. Only the 256 and 1,024 settings constrain Car's context. Sampling time is not comparable with the process wall times in Figure 3.

D.3 REPRESENTATION PROBES

We fit linear ridge probes to query embeddings at the feature encoder and after each of the 16 transformer layers. For each of eight synthetic function families, probes are fitted on 56 task functions and evaluated on 24 held-out task functions. Each task has 256 context rows, 32 query rows, and four input features. Query features are shared across tasks, and query targets are replaced with zero. We compare paired context with a control that shuffles targets within each context, fitting separate probes on the same task split.

Figure 8 reports held-out R^2 with ridge strength 10 at diffusion time $t = 0$. In all eight families, late-layer paired-context scores are positive and exceed the shuffled controls; encoder scores are near zero. The higher scores with correctly paired context indicate that the representations encode information about the task's feature–target relationship. This experiment uses one task seed and a fixed zero-masked query target.

Peak held-out R^2 over layers ranges from 0.83 on linear functions and 0.78 on single-hidden-layer MLPs to 0.30 on depth-3 and depth-4 trees, in every case at layer 11 or later. The shuffled-context controls peak near or below zero. At the encoder the paired and shuffled scores are identical to machine precision, which is the expected degenerate case: before any row attention the query embeddings do not depend on the context at all.

Table 9: Downstream AUROC after target-table training from the pretrained checkpoint and from random initialization, for all 45 matched settings. Each dataset/rate/mode pair is one training trajectory, so the five update budgets are correlated checkpoints. Negative differences are shown in the two settings where they occur.

Dataset	Learning rate	Updates	Pretrained	Scratch	Difference
Adult	2×10^{-5}	100	0.9059	0.4580	+0.4479
Adult	2×10^{-5}	500	0.9087	0.6989	+0.2098
Adult	2×10^{-5}	1,000	0.9080	0.8610	+0.0470
Adult	2×10^{-5}	2,500	0.9094	0.8955	+0.0139
Adult	2×10^{-5}	5,000	0.9102	0.8994	+0.0108
Adult	10^{-4}	100	0.9069	0.4782	+0.4288
Adult	10^{-4}	500	0.9072	0.8659	+0.0413
Adult	10^{-4}	1,000	0.9098	0.9002	+0.0096
Adult	10^{-4}	2,500	0.9118	0.9071	+0.0047
Adult	10^{-4}	5,000	0.9114	0.9078	+0.0035
Adult	3×10^{-4}	100	0.9072	0.4516	+0.4556
Adult	3×10^{-4}	500	0.9090	0.8550	+0.0540
Adult	3×10^{-4}	1,000	0.9117	0.9063	+0.0054
Adult	3×10^{-4}	2,500	0.9121	0.9088	+0.0033
Adult	3×10^{-4}	5,000	0.9140	0.9124	+0.0015
Car	2×10^{-5}	100	0.9640	0.4294	+0.5346
Car	2×10^{-5}	500	0.9878	0.8694	+0.1184
Car	2×10^{-5}	1,000	0.9862	0.8675	+0.1187
Car	2×10^{-5}	2,500	0.9940	0.8955	+0.0985
Car	2×10^{-5}	5,000	0.9888	0.9266	+0.0622
Car	10^{-4}	100	0.9722	0.5404	+0.4318
Car	10^{-4}	500	0.9896	0.8753	+0.1142
Car	10^{-4}	1,000	0.9953	0.9244	+0.0708
Car	10^{-4}	2,500	0.9980	0.9743	+0.0237
Car	10^{-4}	5,000	0.9947	0.9779	+0.0169
Car	3×10^{-4}	100	0.9757	0.6398	+0.3359
Car	3×10^{-4}	500	0.9910	0.9142	+0.0768
Car	3×10^{-4}	1,000	0.9919	0.9779	+0.0140
Car	3×10^{-4}	2,500	0.9916	0.9953	-0.0037
Car	3×10^{-4}	5,000	0.9967	0.9983	-0.0015
Shoppers	2×10^{-5}	100	0.8986	0.3666	+0.5320
Shoppers	2×10^{-5}	500	0.9101	0.5170	+0.3931
Shoppers	2×10^{-5}	1,000	0.9166	0.7559	+0.1607
Shoppers	2×10^{-5}	2,500	0.9186	0.8732	+0.0454
Shoppers	2×10^{-5}	5,000	0.9224	0.8706	+0.0518
Shoppers	10^{-4}	100	0.8894	0.4372	+0.4522
Shoppers	10^{-4}	500	0.9242	0.7796	+0.1446
Shoppers	10^{-4}	1,000	0.9211	0.8807	+0.0405
Shoppers	10^{-4}	2,500	0.9189	0.9063	+0.0126
Shoppers	10^{-4}	5,000	0.9217	0.9185	+0.0032
Shoppers	3×10^{-4}	100	0.9082	0.4393	+0.4689
Shoppers	3×10^{-4}	500	0.9278	0.8238	+0.1039
Shoppers	3×10^{-4}	1,000	0.9246	0.8910	+0.0335
Shoppers	3×10^{-4}	2,500	0.9251	0.9122	+0.0129
Shoppers	3×10^{-4}	5,000	0.9269	0.9125	+0.0145

Table 10: Complete context/steps sweep: one recorded run per configuration. CDE, PCC, C2ST, α and β are scaled by 100. Context is the requested cap; Car has 1,555 training rows, so its 4,096 setting uses all of them. Time is measured inside the generation routine and excludes checkpoint loading.

Dataset	Context	Steps	CDE	PCC	C2ST	α	β	AUROC	Time (s)
Adult	256	10	97.11	91.52	85.37	98.63	37.57	0.8955	14.82
Adult	256	50	98.31	93.17	91.22	95.88	38.19	0.8947	56.48
Adult	256	100	98.39	93.32	91.39	95.42	38.04	0.8964	108.97
Adult	1,024	10	97.55	94.56	89.82	97.31	42.82	0.9005	19.40
Adult	1,024	50	98.78	96.18	95.14	99.02	42.55	0.8971	76.27
Adult	1,024	100	98.87	96.26	95.92	98.72	42.34	0.8951	147.35
Adult	4,096	10	97.48	94.79	89.43	96.77	43.13	0.9004	43.05
Adult	4,096	50	98.77	96.61	96.24	99.28	43.16	0.8976	160.30
Adult	4,096	100	98.83	96.65	95.79	99.09	42.95	0.8982	306.75
Car	256	10	97.30	95.02	99.03	97.91	99.57	0.9396	0.36
Car	256	50	97.12	94.50	99.82	97.83	99.54	0.9347	1.71
Car	256	100	97.15	94.50	97.38	97.73	99.67	0.9428	3.38
Car	1,024	10	98.28	97.23	100.00	97.81	99.60	0.9250	0.53
Car	1,024	50	98.14	97.14	100.00	98.33	99.55	0.9303	2.40
Car	1,024	100	98.12	97.14	100.00	98.32	99.56	0.9354	4.73
Car	4,096	10	98.20	97.17	100.00	96.88	99.50	0.9346	0.66
Car	4,096	50	98.22	96.98	99.55	97.93	99.55	0.9282	2.87
Car	4,096	100	98.24	97.07	100.00	98.06	99.65	0.9337	5.64
Shoppers	256	10	95.11	92.51	75.26	91.44	38.65	0.8880	4.26
Shoppers	256	50	97.22	93.62	92.22	98.40	36.15	0.8885	20.09
Shoppers	256	100	97.40	93.53	92.97	98.56	35.44	0.8926	39.85
Shoppers	1,024	10	95.12	92.69	74.37	91.62	38.90	0.8922	6.13
Shoppers	1,024	50	97.61	94.00	92.08	98.12	35.50	0.8988	27.58
Shoppers	1,024	100	97.79	94.14	95.27	98.18	35.05	0.8864	54.40
Shoppers	4,096	10	94.79	92.55	72.45	90.77	38.48	0.8846	14.65
Shoppers	4,096	50	97.36	93.90	91.92	97.55	36.19	0.8843	57.83
Shoppers	4,096	100	97.56	93.95	93.90	97.96	35.70	0.8965	111.81