GRAFT: A LIGHTWEIGHT SHARED GENERATIVE BACKBONE FOR EFFICIENT CROSS-SCHEMA TABULAR SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data underpins decision-making in finance, healthcare, and business intelligence, yet many real-world tables are small, imbalanced, or access-restricted. Predictive foundation models for tabular data have matured rapidly, and a first generation of *generative* tabular foundation models has emerged. These generators, however, are built on diffusion, transformer, or language-model backbones that are expensive to adapt to a new schema and slow to sample. We ask whether a far lighter design can play the same role where synthetic data matters most. We propose GRAFT, a cross-schema generative model that decouples universal tabular structure from schema-specific detail. A single sub-million-parameter generator, conditioned on a fixed-length meta-feature vector describing the target table, is shared across all schemas. Lightweight per-schema mapping layers project its output into each feature space, and per-schema local discriminators supply specialized feedback without gradient interference. After pretraining on a corpus of heterogeneous public tables, the generator is frozen and adapts to an unseen schema by training only a new mapping layer and discriminator for ten epochs. Under a strict held-out protocol over 86 unseen schemas and three downstream evaluators, GRAFT improves augmented-classifier accuracy over from-scratch CTGAN and TVAE trained for the same or thirty times larger budgets (Wilcoxon $p < 10^{-3}$ in all cases). It also matches the augmentation utility of TabPFGen, a training-free generator built on a pretrained transformer an order of magnitude larger, while producing rows in a single forward pass rather than an iterative Langevin loop. The advantage is concentrated where it matters most: on the smallest fifth of tables GRAFT leads every baseline, including TabPFGen. We also quantify the price of this design, an augmentation-versus-fidelity trade-off.

1 INTRODUCTION

Tabular data is the dominant representation of information in finance, healthcare, and business intelligence, yet many real-world tables are small, imbalanced, or access-restricted. Synthetic data generation is a practical response: it provides substitutes that can *augment* scarce training sets and be shared with fewer restrictions. Unlike text or images, however, tables are heterogeneous, lack spatial structure, and share no vocabulary across schemas. For years this resisted the foundation-model paradigm. That has changed on the *predictive* side: prior-data-fitted and in-context-learning models now generalize across unseen classification and regression tasks without per-dataset training (Hollmann et al., 2023; 2025; Qu et al., 2025; Ma et al., 2025; Kong & Das, 2026). The *generative* side is younger but no longer empty. A first wave of generative tabular foundation models has appeared: latent-diffusion cross-table synthesizers (Lin et al., 2025), diffusion-transformer generators built on pretrained tabular representations (Jiang et al., 2026), energy-based generators derived from TabPFN (Ma et al., 2023), and language-model-based synthesizers (Borisov et al., 2023; Xu et al., 2024). These works establish that cross-table generative pretraining is feasible, but they share two practical costs: a large backbone (a diffusion model and decoder, an autoencoder over a unified latent space, a language model, or a transformer that attends over the entire training table), and iterative sampling. For the small tables where augmentation is most needed, both costs are hard to justify. A practitioner who can run a multi-million-parameter transformer through hundreds of sampling steps

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.

per batch could just as well train an ensemble of specialized models. This leaves an opening for a *lightweight* generative backbone that is cheap to adapt and fast to sample, which is the contribution we pursue.

Generative adversarial networks (GANs) are a natural substrate for this goal. A GAN generates a sample in a single forward pass. Specialized architectures such as CTGAN (Xu et al., 2019) already handle the discrete, non-Gaussian, multi-modal nature of tabular features through mode-specific normalization and training-by-sampling. Their limitation is that they are single-table: every new schema requires a full training cycle and transfers nothing. We evolve this framework into a cross-schema generative backbone, GRAFT (**G**enerative **R**apid **A**daptation **F**or **T**abular data). A single generator, conditioned on a fixed-length meta-feature vector describing the target dataset, learns structure shared across tables. Lightweight per-schema mapping layers align its fixed-size output to each feature space, and per-schema local discriminators provide specialized feedback. Adapting to an unseen schema trains only a new mapping layer and discriminator while the generator stays frozen: a small head and a handful of epochs, in the spirit of parameter-efficient transfer (Houlsby et al., 2019; Hu et al., 2022), here applied to a generative rather than a discriminative model.

We are explicit about where this design wins and where it does not. Because the shared generator regresses toward a cross-dataset prior, it emits "typical" samples that broaden coverage and help augmentation. It reproduces dataset-specific tail structure less faithfully than a single-table model, so a dedicated real-versus-synthetic classifier can still distinguish its samples. Our strengths therefore concentrate in augmentation utility, adaptation cost, and sampling speed; the fidelity trade-off is quantified in Section 4.6.

Contributions.

- **A lightweight cross-schema generative architecture** (Section 3): one meta-feature-conditioned shared generator of roughly 0.4M parameters, per-schema mapping layers, and per-schema local discriminators. A new schema costs a small head trained for a few epochs, and a generated row costs one forward pass.
- **Augmentation utility at low cost** (Sections 4.3 and 4.5): after ten adaptation epochs GRAFT significantly outperforms CTGAN and TVAE trained for the same budget and for $30\times$ longer. It is statistically indistinguishable from TabPFGen, a training-free generator with a $10\times$ larger transformer backbone and iterative sampling.
- **A small-data advantage and an honest trade-off** (Sections 4.4 and 4.6): gains grow as tables shrink, and on the smallest quintile GRAFT leads every baseline. The same cross-dataset prior costs fidelity, which we measure with ML-efficacy and C2ST and explain.

2 Related Work

Tabular data remains a challenging modality: tree ensembles frequently match or beat deep networks because of feature heterogeneity and the absence of spatial invariance (Borisov et al., 2024; Grinsztajn et al., 2022). We organize prior work into single-table generators, meta-learning with dataset descriptors, and tabular foundation models. We distinguish predictive from generative models because that distinction defines our contribution.

Single-table generative models. Early tabular synthesis captured the joint distribution of one dataset in isolation. CTGAN (Xu et al., 2019) introduced a conditional generator and training-by-sampling to handle multi-modal continuous columns and imbalanced categories; TVAE (Xu et al., 2019) is its variational counterpart. GAN-based synthesis has since been extended to specialized domains (Bourou et al., 2021) and strengthened architecturally (e.g., CTAB-GAN+, Zhao et al., 2024). Diffusion models then raised the fidelity ceiling: TabDDPM (Kotelnikov et al., 2023) and latent score-based models such as TabSyn (Zhang et al., 2024) are today's strongest single-table synthesizers on standard machine-learning-efficiency benchmarks. Evaluation has matured in parallel, spanning resemblance, utility, and privacy (Hernadez et al., 2023; Alaa et al., 2022; Yang et al., 2024). All of these generators, however, are *schema-locked*: each requires a complete training cycle per table and transfers nothing across tables.

Meta-learning with dataset descriptors. Deep tabular architectures such as TabNet (Arik & Pfister, 2021) and FT-Transformer (Gorishniy et al., 2021) are hard to tune. This spurred AutoML (He

et al., 2021; Hutter et al., 2019), architecture search (Elsken et al., 2019), and—most relevant here—meta-learning (Brazdil et al., 2022), which learns across datasets via meta-features (Pinto et al., 2016) and pretrained selectors such as RankML, AutoGRD, and Dataset2Vec (Dagan et al., 2024; Laadan et al., 2019; Cohen-Shapira et al., 2019; Jomaa et al., 2021). These methods share our use of dataset-level descriptors but are *discriminative*: they select or configure a predictor. We repurpose the idea to *condition a generator*.

Tabular foundation models: predictive versus generative. The foundation-model paradigm (Bommasani et al., 2021; Zhou et al., 2023) reached tabular data first on the predictive side (van Breugel & van der Schaar, 2024). TabPFN (Hollmann et al., 2023) performs in-context learning without parameter updates; TabPFN v2 (Hollmann et al., 2025), TabICL (Qu et al., 2025), TabDPT (Ma et al., 2025), and TabFM (Kong & Das, 2026) scale this to larger tables. These models eliminate the cold start for *prediction* but do not, by themselves, synthesize data. The generative counterpart is newer. TabPFGen (Ma et al., 2023) turns TabPFN into an energy-based generator: the class-conditional energy is read off TabPFN's logits, and rows are sampled with stochastic gradient Langevin dynamics. Each Langevin step is a forward-and-backward pass through the transformer conditioned on the entire training table. TabPFGen needs no training, which makes it the natural strong baseline for our setting, and we compare against it directly. CTSyn (Lin et al., 2025) consolidates heterogeneous tables into a unified latent space and generates with a schema-conditioned latent *diffusion* model; TabFORGE (Jiang et al., 2026) pretrains a diffusion transformer and finetunes it, with its decoder, per table; LLM-based generators (Borisov et al., 2023; Xu et al., 2024) synthesize rows from language backbones. Relative to this line, GRAFT differs in three ways that matter for deployment. (i) It is *GAN-based*, so generation is a single forward pass rather than a denoising or Langevin loop. (ii) Adaptation *freezes* a sub-million-parameter shared generator and trains only a small per-schema head; it does not fine-tune a diffusion model, an autoencoder, or a language model, and it does not attend over the full training table at every sampling step. (iii) It uses *per-schema local discriminators*, which avoid the gradient interference a single global critic incurs across many tables.

3 METHOD

3.1 OVERVIEW

GRAFT (Figure 1) learns universal structural patterns across heterogeneous schemas and synthesizes data for unseen tables after a lightweight adaptation. Unlike a GAN constrained to one dataset's dimensionality, it has three components. (1) Each dataset D_i is summarized by a fixed-size meta-feature vector M_i, so that the shared *universal generator* G_θ can condition on any dataset regardless of its size and produce a fixed-size latent output. (2) Per-schema *mapping layers* $L_1, \ldots, L_n$ project that output into the feature space of each target dataset. (3) Per-schema *local discriminators* $LD_1, \ldots, LD_n$ critique the generated samples of their respective datasets. This separation decouples general tabular logic (joint distributions, feature correlations) from the structural constraints of individual tables. It has three intended advantages. **Scale as a regularizer**: diverse data across domains pushes the generator toward shared tabular behavior rather than one table. **Specialized feedback**: local discriminators, rather than one global critic, avoid gradient interference. **Efficient adaptation**: a new schema costs a small head trained for a few epochs, which sidesteps the cold start where from-scratch GANs converge slowly or unstably on small tables. (Adaptation uses the full new table for a few epochs with few trainable parameters; it is parameter- and epoch-efficient, not few-shot.)

3.2 ARCHITECTURE

Universal generator G_θ. The core is a shared generator built on a modified CTGAN. The design hypothesis is that tabular data, despite domain diversity, shares a common "syntax": recurring ways in which numerical modes co-vary with categorical distributions. The generator takes the concatenation of a latent noise vector z and a meta-feature vector M_i as input. It thus learns a conditional mapping from a statistical fingerprint to plausible structure rather than memorizing one table. In our implementation $z \in \mathbb{R}^{128}$ and $M_i \in \mathbb{R}^{17}$ (a 145-dimensional input); the body is three fully con-

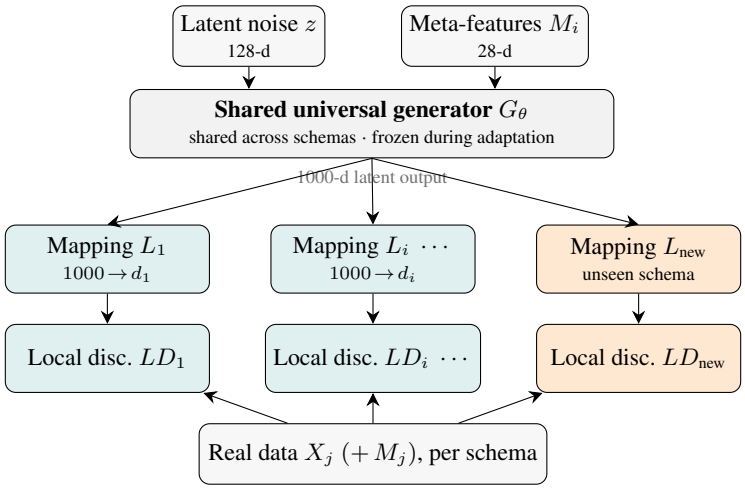

Figure 1: Overview of GRAFT. A single shared generator G_θ, conditioned on latent noise z and a fixed-length meta-feature vector M_i, produces a schema-agnostic latent that per-schema mapping layers L_j project into each feature space; per-schema local discriminators LD_j compare synthetic against real samples X_j (both concatenated with M_j). During adaptation to an unseen schema, G_θ is frozen (gray) and only a new L_{new} and LD_{new} are trained (orange).

nected layers of width 256; the output is a fixed-size 1,000-dimensional latent vector shared across all schemas. The generator has roughly 0.43M parameters in total.

Per-schema mapping layers $L_1, \ldots, L_n$. Datasets differ in dimensionality (one table may have 10 columns, another 200). Each L_i is a small trainable module that projects the generator's 1,000-dimensional output into the target feature space. It compresses $1{,}000 \to 256$ and then projects to the d_i (encoded) features of D_i. Offloading dimensional alignment to these layers preserves the core generator for universal distributional logic. During adaptation, L_{new} and LD_{new} are the only trained parameters.

Per-schema local discriminators $LD_1, \ldots, LD_n$. A single global discriminator over many tables suffers gradient interference: forced to model all joint distributions at once, it returns a blurred, averaged signal. We instead assign each dataset its own discriminator LD_i, responsible only for the authenticity of D_i. The generator thus receives feedback specific to that domain's correlations and marginals while retaining its global perspective. Each LD_i takes the concatenation of a (real or synthetic) sample and M_i; its body is three layers of width 256 with LeakyReLU (slope 0.2) and dropout 0.5. The mapping layer and local discriminator trained for a new schema together amount to $\approx 0.4\text{M} + 513\, d_i$ parameters. At sampling time only G_θ and L_{new} (≈ 0.7M parameters for typical d_i) are exercised.

3.3 META-FEATURES

We summarize each dataset D_i as a fixed-length vector $M_i \in \mathbb{R}^{28}$, so the generator receives structural context independent of the number of columns. The 28 dimensions form six families. **Global statistics** (3): number of samples, features, and target classes. **Feature composition** (5): number of continuous and of categorical columns, their two ratios, and minority-class percentage. **Continuous distributional moments** (4): min, max, mean, and std aggregated across continuous columns. **Discrete distributional moments** (4): min, max, mean, and std of unique-value counts across categorical columns. **Relational moments** (4): min, max, mean, and std of per-feature feature–target χ^2 association, with continuous features equal-width-discretized first. **Distribution-fit moments** (8): for each continuous column, the KL divergence between its empirical histogram and a fitted normal distribution, and separately a fitted log-normal distribution; each set of per-column divergences is aggregated into min, max, mean, and std. Aggregating every per-column statistic into four moments yields a length-invariant descriptor (Section A).

3.4 TRAINING AND ADAPTATION

Training proceeds in two phases: pretraining across schemas and efficient adaptation to a new one. Let $\hat{x} = L_i(G_\theta(z, M_i))$ denote a synthetic sample for dataset D_i with $z \sim \mathcal{N}(0, I)$, and let $x \sim D_i$ denote a real sample. Following CTGAN, we optimize a Wasserstein objective with gradient penalty (Gulrajani et al., 2017). Each local discriminator minimizes

$$\mathcal{L}_{LD_i} = \mathbb{E}_z\big[LD_i(\hat{x}, M_i)\big] - \mathbb{E}_{x\sim D_i}\big[LD_i(x, M_i)\big] + \lambda\,\mathbb{E}_{\tilde{x}}\Big[\big(\|\nabla_{\tilde{x}} LD_i(\tilde{x}, M_i)\|_2 - 1\big)^2\Big], \quad (1)$$

where $\tilde{x}$ is sampled on interpolants between real and synthetic samples and λ is the gradient-penalty weight. For dataset D_i the generator, together with its mapping layer L_i, minimizes

$$\mathcal{L}_{G,i} = -\,\mathbb{E}_z\big[LD_i(L_i(G_\theta(z, M_i)), M_i)\big] \;+\; \beta\,\mathcal{L}_i^{\text{cond}}, \quad (2)$$

where $\mathcal{L}_i^{\text{cond}}$ is CTGAN's conditional cross-entropy term, which enforces the sampled condition vector on the mapped output, and β is its weight.

During **pretraining**, datasets are visited in round-robin order. At each step we draw D_i, update LD_i and L_i, and update the shared generator with $\nabla_\theta \mathcal{L}_{G,i}$. Over an epoch the shared parameters therefore receive $\sum_i \nabla_\theta \mathcal{L}_{G,i}$, one dataset per step, with each gradient sourced from a *specialist* discriminator rather than from the averaged, interfering signal of a single global critic. During **adaptation** to an unseen dataset D_{new}, we freeze G_θ and train only a freshly initialized L_{new} and LD_{new}. Because the generator already encodes universal structure, a few epochs suffice. All hyperparameters are listed in Section B.

4 EXPERIMENTAL EVALUATION

Our evaluation is organized around the axes on which a lightweight cross-schema generator should help: augmentation utility (especially with little data), adaptation and sampling cost, and generalization to unseen schemas. Fidelity is a secondary axis with an explicit trade-off analysis. The claim is not the most realistic tables, but the best augmentation quality per unit of compute in the small-data regime, with single-pass sampling.

4.1 SETUP AND HELD-OUT PROTOCOL

Data. The corpus consists of 86 public classification datasets exported from OpenML (Vanschoren et al., 2014) and KEEL (Derrac et al., 2015), spanning finance, healthcare, biology, and the physical and social sciences, with 540 to 9,961 rows, 2 to 256 features, 2 to 10 classes, and mixed numerical and categorical columns (Section C). Following CTGAN, continuous features use mode-specific normalization via a variational Gaussian mixture, and categorical features use one-hot encoding with training-by-sampling.

Held-out protocol. All reported metrics are computed on schemas that were *never* seen by the generator during pretraining. We use 4-fold cross-validation over datasets. The corpus is partitioned into four disjoint folds of 24, 22, 20, and 20 datasets. In each round the shared generator is pretrained on the three remaining folds (62–66 datasets), and each dataset of the held-out fold is adapted and evaluated as an unseen schema. Every dataset is thus evaluated exactly once, and every number in the paper is a mean or a paired statistic over these 86 held-out schemas. Within each held-out dataset, rows are split 75%/25% into a training partition and a test partition. The training partition is used for adaptation, for the from-scratch baselines, as the in-context table of the training-free baseline, and as the real portion of augmented training sets. The test partition is used only for evaluation and is never seen by any generator, encoder, or classifier before it.

Training and budgets. The shared generator is pretrained for 50 epochs, visiting all pretraining datasets in round-robin order each epoch. We use Adam (learning rate 2×10^{-4}, weight decay 10^{-6}) for generator and discriminators and a batch size of 100. For adaptation the generator is frozen and a new mapping layer and discriminator are trained. To make cost explicit, we evaluate every trainable method at eight checkpoints: 1, 5, 10, 20, 50, 100, 200, and 300 epochs. At each checkpoint the from-scratch baselines receive exactly as many training epochs on the same training partition as GRAFT receives adaptation epochs. Headline numbers are reported at the **10-epoch budget** we recommend in practice; the longer budgets show that the conclusions do not depend on stopping early.

4.2 BASELINES, EVALUATORS, AND METRICS

Baselines. We compare against three generators. All methods see the same training partition and are evaluated on the same test partition, and the number of synthetic rows equals the training-partition size.

- **CTGAN** (Xu et al., 2019), trained from scratch. This is the natural test of whether a pretrained backbone provides a better inductive bias than the same GAN family with no prior.
- **TVAE** (Xu et al., 2019), a variational autoencoder from the same library and the stronger of the two on fidelity.
- **TabPFGen** (Ma et al., 2023), the foundation-model baseline. It requires no training, but every generated batch runs stochastic gradient Langevin dynamics through a pretrained TabPFN transformer (Hollmann et al., 2023; 2025) that attends over the full training partition at each step.

Downstream evaluators. Utility is measured with three predictors that cover the dominant tabular model families: two tree ensembles, **XGBoost** (Chen & Guestrin, 2016) and **Random Forest** (Breiman, 2001), and a margin-based learner, **SVM** (Cortes & Vapnik, 1995). Conclusions therefore do not hinge on one model class.

Metrics. We report four metrics per held-out schema. **Augmentation utility** (primary): the real training partition is augmented with an equal number of synthetic rows, and a predictor is trained on the union and evaluated on the real test partition. We report accuracy and AUC, and against a baseline the paired difference $\Delta = \text{Acc}_{\text{ours}} - \text{Acc}_{\text{baseline}}$ (higher is better). **ML-efficacy** (train-on-synthetic, test-on-real): a predictor trained *only* on synthetic rows is evaluated on real test rows (higher is better). **C2ST**: the accuracy of a classifier trained to distinguish real from synthetic rows (Lopez-Paz & Oquab, 2017); 0.5 is ideal and *lower is better*. **Cost**: parameters trained per schema, parameters exercised at sampling, and the sampling procedure (Section 4.5). A dataset is a *win* for A over B when $\Delta > 0$. Since a bare win rate says nothing about magnitude or ties, we also report the median Δ with its inter-quartile range and a two-sided paired Wilcoxon signed-rank test across schemas (Wilcoxon, 1945; Demšar, 2006).

4.3 MAIN RESULTS: AUGMENTATION UTILITY ON UNSEEN SCHEMAS

Table 1 summarizes all metrics at the 10-epoch budget, averaged over the 86 held-out schemas, and Table 2 gives the paired statistics. The same tables at 300 epochs and at every checkpoint are in Section D and tell the same story. Three findings stand out.

Augmentation helps, and GRAFT helps most. Every generator improves over training on real rows alone, and GRAFT has the highest mean augmentation accuracy for all three evaluators ($+0.115$, $+0.041$, and $+0.118$ over real-only for XGBoost, SVM, and Random Forest; $p < 10^{-11}$). **Against the same GAN family without a prior, the gain is large and significant.** GRAFT beats CTGAN on 65%, 87%, and 65% of schemas (ties on a further 17%, 3%, and 13%), with mean Δ of $+0.010$, $+0.039$, and $+0.010$ and Wilcoxon $p < 10^{-4}$ for every evaluator. The numbers against TVAE are nearly identical. The larger SVM gain reflects that a margin learner benefits most from added coverage of the distribution's bulk. **Against the foundation-model baseline, GRAFT is on par.** TabPFGen is a strong generator: it has the best ML-efficacy and C2ST of all methods. Yet on augmentation utility its paired difference to GRAFT is not significant for any evaluator ($p = 0.60$, 0.64, 0.43). The median Δ is exactly zero and the mean is slightly in our favor ($+0.002$, $+0.015$, $+0.004$). GRAFT also has the highest augmentation AUC for XGBoost and Random Forest; TabPFGen leads on SVM AUC (Table 11). GRAFT reaches this parity with a $\sim$0.7M-parameter sampler in one forward pass. We therefore regard the tie as the paper's central result: the augmentation utility of a large in-context transformer at a fraction of its cost.

Figure 2 traces the three trainable methods across budgets. GRAFT is ahead of CTGAN and TVAE from the very first epoch, and its augmentation accuracy is essentially flat thereafter. This is the signature of a transferred prior: the head has little to learn about the shared structure and only needs to align dimensions. The from-scratch models also plateau early on augmentation, so the equal-budget gap never closes (Table 12). The per-schema box plots in Figure 3 (Section D) show a shift of the whole distribution rather than a few outliers.

Table 1: Held-out results at the 10-epoch budget (CTGAN and TVAE trained from scratch for 10 epochs; TabPFGen is training-free), mean over 86 unseen schemas. Augmentation and ML-efficacy: higher is better; C2ST: 0.5 is ideal, lower is better. Best generator per row in bold.

Evaluator	Metric	Real only	CTGAN	TVAE	TabPFGen	GRAFT
XGBoost	Augmentation acc. (↑)	0.852	0.957	0.959	0.965	**0.967**
	Augmentation AUC (↑)	0.909	0.983	0.979	0.980	**0.986**
	ML-efficacy acc. (↑)	–	0.427	0.521	**0.727**	0.442
	C2ST acc. (↓)	–	0.874	0.876	**0.814**	0.983
SVM	Augmentation acc. (↑)	0.778	0.780	0.781	0.803	**0.818**
	Augmentation AUC (↑)	0.836	0.840	0.867	**0.886**	0.868
	ML-efficacy acc. (↑)	–	0.440	0.284	**0.585**	0.446
	C2ST acc. (↓)	–	0.784	0.792	**0.645**	0.897
Random Forest	Augmentation acc. (↑)	0.853	0.961	0.961	0.967	**0.971**
	Augmentation AUC (↑)	0.909	0.988	0.980	0.982	**0.990**
	ML-efficacy acc. (↑)	–	0.435	0.518	**0.727**	0.410
	C2ST acc. (↓)	–	0.842	0.859	**0.790**	0.972

Table 2: Paired augmentation-accuracy statistics of GRAFT against each baseline across 86 held-out schemas at the 10-epoch budget. Win/tie/loss: fractions of schemas with $\Delta > 0$, $= 0$, < 0; p: two-sided Wilcoxon signed-rank test (Pratt zero handling).

Evaluator	GRAFT vs.	Win	Tie	Loss	Mean Δ	Median Δ [IQR]	Wilcoxon p
XGBoost	CTGAN	65%	17%	17%	$+0.010$	$+0.003$ [+0.000, +0.017]	$<10^{-4}$
	TVAE	65%	14%	21%	$+0.008$	$+0.003$ [+0.000, +0.012]	$<10^{-4}$
	TabPFGen	37%	16%	47%	$+0.002$	$+0.000$ [-0.004, +0.004]	0.597
SVM	CTGAN	87%	3%	9%	$+0.039$	$+0.025$ [+0.008, +0.069]	$<10^{-4}$
	TVAE	84%	3%	13%	$+0.037$	$+0.024$ [+0.006, +0.063]	$<10^{-4}$
	TabPFGen	44%	10%	45%	$+0.015$	$+0.000$ [-0.012, +0.017]	0.639
Random Forest	CTGAN	65%	13%	22%	$+0.010$	$+0.005$ [+0.000, +0.014]	$<10^{-4}$
	TVAE	66%	10%	23%	$+0.010$	$+0.004$ [+0.000, +0.016]	$<10^{-4}$
	TabPFGen	40%	21%	40%	$+0.003$	$+0.000$ [-0.003, +0.006]	0.430

4.4 Small-data regime

The pretrained prior should matter most when local data is scarce, and it does. Table 3 stratifies the held-out schemas by size. The gain over CTGAN grows monotonically as N shrinks: for XGBoost, from $+0.003$ at $N > 5{,}000$ to $+0.037$ at $N \leq 1{,}000$. The same pattern holds for SVM and Random Forest (Tables 16 and 17). More telling is the comparison with TabPFGen. On the large schemas the two are indistinguishable (XGBoost $|\Delta| < 0.001$ at $N > 5{,}000$). On the smallest quintile, the 17 held-out schemas with $N \leq 1{,}156$ that we define as the *small-data regime*, GRAFT leads every baseline on every evaluator: $+0.023$/$+0.019$/$+0.012$ over CTGAN/TVAE/TabPFGen for XGBoost (win rates 82%/65%/65%; Wilcoxon $p = 0.001$/0.018/0.052), $+0.043$/$+0.042$/$+0.046$ for SVM, and $+0.022$/$+0.022$/$+0.012$ for Random Forest. With 17 schemas the TabPFGen comparison is at the edge of significance, but its direction is consistent across evaluators and budgets. This is the regime a cheap, well-regularized prior should win. A few hundred rows are too few for a from-scratch model to fit and for an in-context transformer to condition on, whereas the shared generator brings structure learned from more than sixty other tables.

4.5 Adaptation cost and sampling speed

The central practical advantage of GRAFT is cost, and Table 4 makes the accounting explicit. Adapting to a new schema trains only a mapping layer and a local discriminator ($\approx 0.4\text{M} + 513\, d_i$ parame-

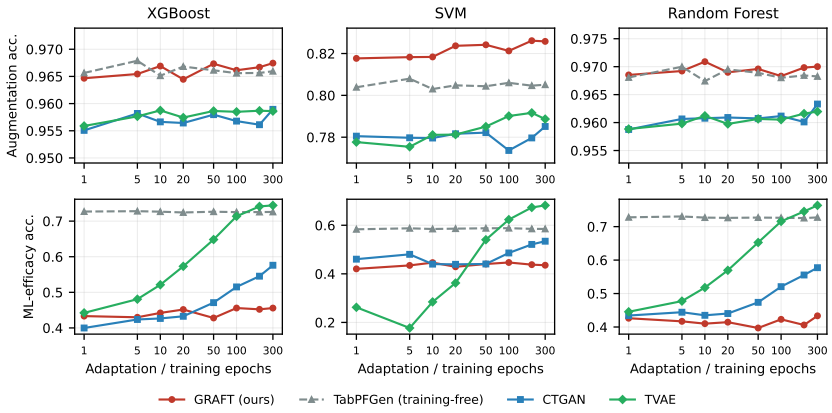

Figure 2: Augmentation accuracy (top) and ML-efficacy (bottom) across adaptation/training epochs (log scale), mean over 86 held-out schemas. GRAFT leads the from-scratch models on augmentation from the first epoch at every budget; training-free TabPFGen (dashed) is constant. On ML-efficacy the ordering reverses and the from-scratch models keep improving, the trade-off analyzed in Section 4.6.

Table 3: Augmentation accuracy (XGBoost) by held-out dataset size at the 10-epoch budget. The smallest-quintile row (17 schemas) is the small-data regime of the text; the other rows partition all 86 schemas. Best generator per row in bold.

Held-out size	**#**	**Real only**	**CTGAN**	**TVAE**	**TabPFGen**	**GRAFT**	**Δ vs. CTGAN / TabPFGen**
Smallest quintile ($N \leq 1,156$)	17	0.791	0.930	0.934	0.941	**0.953**	+0.023 / +0.012
$N \leq 1{,}000$	7	0.714	0.902	0.906	0.918	**0.939**	+0.037 / +0.021
$1{,}000 < N \leq 2{,}000$	39	0.816	0.948	0.951	0.959	**0.959**	+0.011 / +0.001
$2{,}000 < N \leq 5{,}000$	27	0.902	0.971	0.972	**0.977**	0.976	+0.006 / −0.001
$N > 5{,}000$	13	0.930	0.982	0.983	**0.985**	0.985	+0.003 / +0.000

ters) with the generator frozen, for ten epochs over the training partition. CTGAN and TVAE train a comparably sized model from scratch. Against them the saving is therefore not parameters; it is that the same small budget is spent on top of a transferred prior, which is what the equal-budget curves in Figure 2 measure. Table 9 (Section D) pushes this with a deliberately unfair comparison: GRAFT after 5 or 10 adaptation epochs against CTGAN and TVAE after 300 training epochs, thirty to sixty times the compute. GRAFT still wins significantly on every evaluator ($p < 10^{-3}$ throughout; e.g., $+0.011$ and $+0.039$ mean Δ over CTGAN@300 for XGBoost and SVM).

Against TabPFGen the accounting is different in kind. TabPFGen trains nothing, but every generated batch costs η Langevin steps. Each step is a forward and backward pass through a pretrained TabPFN transformer ($\approx$ 7M parameters for v2, $\approx$ 26M for v1) whose attention spans the *entire* training table concatenated with the batch, so per-step cost grows with the table and the number of features. GRAFT samples a row with one forward pass through G_θ and L_{new}, roughly 0.7M parameters, and never revisits the training data. In other words, the practitioner who can afford TabPFGen's sampler could equally afford an ensemble of dedicated models. GRAFT targets the practitioner who cannot, and Table 1 shows that this practitioner gives up nothing on augmentation utility.

4.6 The augmentation-versus-fidelity trade-off

On ML-efficacy and C2ST, GRAFT trails every baseline (Table 1), and the ML-efficacy gap to the from-scratch models *widens* with epochs (Figure 2, bottom; Tables 14 and 15). At 10 epochs GRAFT and CTGAN are level on efficacy (0.44 versus 0.43 for XGBoost). By 300 epochs CTGAN reaches 0.58 and TVAE 0.74 while GRAFT stays near 0.46, and its C2ST sits near 0.98 versus 0.85 for CTGAN and 0.81 for TabPFGen. This is a predictable consequence of the design, not a defect.

Table 4: Per-schema cost accounting: parameters updated to serve a new table, parameters a generated row passes through, and the sampling procedure. d_i: encoded feature dimension; N: training rows; η: Langevin steps. From-scratch models use their reference implementations' default widths.

Method	Trained per new schema	Exercised at sampling	Sampling per batch	Train table at sampling
CTGAN	full generator + discriminator, from scratch	generator (two 256-wide residual blocks)	1 forward pass	no
TVAE	full encoder + decoder, from scratch	decoder (two 128-wide layers)	1 forward pass	no
TabPFGen	none (training-free)	TabPFN transformer (≈7–26M)	η forward+backward passes, attending over N+batch rows	yes, at every step
GRAFT	mapping + local discriminator ($\approx 0.4\text{M} + 513\, d_i$); G_θ frozen	$G_\theta + L_{\text{new}}$ (≈ 0.7M)	1 forward pass	no

The shared generator regresses toward a cross-dataset prior, so it emits "typical," broadly plausible samples. That is exactly what helps augmentation, where coverage of the distribution's bulk is valuable and the real rows are still present. It is less able to reproduce the dataset-specific tail structure that a single-table model can overfit given enough epochs, and that an in-context transformer can copy from the table it conditions on. The same generality lets a dedicated classifier flag our samples and limits train-on-synthetic-only efficacy. The property that costs us on fidelity is the property that pays off on augmentation and on cost. A user whose goal is a faithful stand-in for the real table should prefer TVAE or TabPFGen, and we report the fidelity numbers in full rather than only at the early epochs where the gap happens to be smaller.

Robustness checks. The per-fold breakdown (Table 18) shows the same ordering in each of the four held-out rounds. GRAFT's per-schema augmentation accuracy also varies less across the eight budgets than either from-scratch model's (Table 19), so the 10-epoch budget is not a tuned sweet spot.

5 Limitations

The augmentation gains come with a fidelity cost (Section 4.6). We claim neither state-of-the-art fidelity, which diffusion models and in-context transformers hold, nor a formal privacy guarantee. The pretraining corpus (62–66 datasets per fold) is small relative to what "foundation" implies. We isolate the value of pretraining through the equal-budget comparison with CTGAN (the same GAN family with no prior) rather than through component-wise ablations of the meta-feature conditioning and local discriminators, which we leave to future work. The small-data advantage over TabPFGen is consistent in direction but, with 17 schemas, only borderline significant. The compact mapping head may limit very high-dimensional schemas, and only numerical and categorical columns are supported.

6 Conclusion

We asked whether a lightweight design can play the role that diffusion, transformer, and language-model backbones now play as generative foundation models for tabular data. GRAFT answers affirmatively for low-resource settings. A sub-million-parameter, meta-feature-conditioned generator, shared across schemas with per-schema mapping layers and local discriminators, learns transferable tabular structure. It adapts to an unseen schema by training a small head for ten epochs while the generator stays frozen. On 86 unseen schemas this significantly exceeds from-scratch CTGAN and TVAE at equal and thirty-fold budgets, matches a TabPFN-based in-context generator with an order of magnitude more parameters and iterative sampling, and leads all baselines on the smallest tables. The price is a fidelity trade-off that we analyze rather than hide. We position GRAFT not as the first generative tabular foundation model but as the cost-effective one for small-scale settings.

Reproducibility statement

The architecture and objectives are fully specified in Section 3; all hyperparameters are listed in Section B; the 28 meta-features are defined in Section A; the 86 held-out datasets with their fold assignment are listed in Section C; and every table in the paper is computed from per-dataset, per-checkpoint result files by the scripts in the supplementary material.

Ethics statement

This work uses only publicly available datasets and involves no human subjects. Synthetic tabular data can support privacy-preserving data sharing, but it can also leak information about training rows; we make no formal privacy claim. As with any generative model, synthetic data may reproduce biases present in the source tables, and practitioners should audit downstream models accordingly.

AI use statement

Large language model (LLM) tools were used solely to assist with the writing of this manuscript, specifically for language polishing, including improving grammar, clarity, phrasing, and overall readability. These tools were not used to generate research ideas, design or conduct experiments, analyze results, or produce scientific content. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the content of this paper.

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 (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 290–306, 2022.

Sercan Ö. Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35(8), pp. 6679–6687, 2021.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, et al. On the opportunities and risks of foundation models, 2021. arXiv:2108.07258.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *International Conference on Learning Representations (ICLR)*, 2023.

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, 2024.

Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. A review of tabular data synthesis using GANs on an IDS dataset. *Information*, 12(9):375, 2021.

Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Springer, 2nd edition, 2022.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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.

Noy Cohen-Shapira, Lior Rokach, Bracha Shapira, Gilad Katz, and Roman Vainshtein. AutoGRD: Model recommendation through graphical dataset representation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 821–830, 2019.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Itai Dagan, Roman Vainshtein, Gilad Katz, and Lior Rokach. Automated algorithm selection using meta-learning and pre-trained deep convolution neural networks. *Information Fusion*, 105: 102210, 2024.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

J Derrac, S Garcia, L Sanchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Logic Soft Comput*, 17(1):255–287, 2015.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 35, 2022.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Mikel Hernadez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic tabular data evaluation in the health domain covering resemblance, utility, and privacy dimensions. *Methods of Information in Medicine*, 62(S1):e19–e38, 2023.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023.

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.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (eds.). *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019.

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

Hadi S. Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. Dataset2Vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985, 2021.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/, June 2026.

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 (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17564–17579, 2023.

Doron Laadan, Roman Vainshtein, Yarden Curiel, Gilad Katz, and Lior Rokach. RankML: A meta learning-based approach for pre-ranking machine learning pipelines, 2019. arXiv:1911.00108.

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

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations (ICLR)*, 2017.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – tabular data generation with TabPFN. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Fábio Pinto, Carlos Soares, and João Mendes-Moreira. Towards automatic generation of metafeatures. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 9651 of *Lecture Notes in Computer Science*, pp. 215–226. Springer, 2016.

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 (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847, 2025.

Boris van Breugel and Mihaela van der Schaar. Position: Why tabular foundation models should be a research priority. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48976–48993, 2024.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

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

Shengzhe Xu, Cho-Ting Lee, Mandar Sharma, Raquib Bin Yousuf, Nikhil Muralidhar, and Naren Ramakrishnan. Are LLMs naturally good at synthetic tabular data generation?, 2024. arXiv:2406.14541.

Scott Cheng-Hsin Yang, Baxter Eaves, Michael Schmidt, Ken Swanson, and Patrick Shafto. Structured evaluation of synthetic tabular data, 2024. arXiv:2403.10424.

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 *International Conference on Learning Representations (ICLR)*, 2024.

Zilong Zhao, Aditya Kunar, Robert Birke, Hiek Van der Scheer, and Lydia Y. Chen. CTAB-GAN+: Enhancing tabular data synthesis. *Frontiers in Big Data*, 6:1296508, 2024.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT, 2023. arXiv:2302.09419.

A META-FEATURE DEFINITIONS

Table 5 enumerates the 17 dimensions of M_i. Continuous columns are standardized before computing moments; categorical columns use their raw category counts. For the relational family, continuous features are equal-width-discretized before computing the χ^2 statistic with the target.

Table 5: The 17 meta-features that form M_i.

#	Family	Feature	Definition
1	Global	#samples	N (log-scaled)
2	Global	#features	d
3	Global	#classes	number of distinct target values
4	Composition	categorical ratio	d_{cat}/d
5	Composition	minority fraction	fraction of rows in the rarest class
6–9	Continuous moments	min / max / mean / std	of per-column statistics over continuous columns
10–13	Discrete moments	min / max / mean / std	of per-column unique-value counts over categorical columns
14–17	Relational moments	min / max / mean / std	of per-feature $\chi^2(\text{feature}, \text{target})$ over all columns

B HYPERPARAMETERS

Table 6: Hyperparameters used in all experiments. Settings marked "CTGAN default" follow the reference implementation of Xu et al. (2019), from which the generator body, encoders, and objective are inherited.

Component / setting	Value
Latent noise z	$\mathbb{R}^{128}, \mathcal{N}(0, I)$
Meta-feature vector M_i	$\mathbb{R}^{28}$ (Section A)
Generator body	3 fully connected layers, width 256; output 1,000-d (≈ 0.43M parameters)
Mapping layer L_i	$1{,}000 \to 256 \to d_i$
Local discriminator LD_i	input $d_i + 28$; 3 layers, width 256; LeakyReLU(0.2); dropout 0.5
Adversarial objective	WGAN-GP; $\lambda = 10$, pac $= 10$, $n_{\text{critic}} = 1$ (CTGAN defaults)
Conditional loss weight β	1 (CTGAN default)
Optimizer	Adam, lr 2×10^{-4}, weight decay 10^{-6}, $(\beta_1, \beta_2) = (0.5, 0.9)$ (CTGAN defaults)
Batch size	100
Pretraining epochs	50 (round-robin over the 62–66 pretraining datasets of each fold)
Adaptation epochs	10 recommended; checkpoints at 1, 5, 10, 20, 50, 100, 200, 300
From-scratch baselines	CTGAN and TVAE with library defaults, trained for the same epochs as the adaptation checkpoint
TabPFGen	reference implementation with its default Langevin settings; training partition as in-context table
Continuous encoding	mode-specific normalization (VGM, 10 modes; CTGAN default)
Categorical encoding	one-hot with training-by-sampling
Downstream evaluators	XGBoost / Random Forest / SVM with library defaults; synthetic rows = training-partition rows
C2ST classifier	the same three evaluators, trained to separate real from synthetic rows on a held-out split

Table 7: The 86 datasets of the corpus, sorted by row count N; d is the number of raw features, C the number of classes, and Fold the held-out round (1–4) in which the schema was adapted and evaluated. File names are the identifiers used in our harness (OpenML and KEEL exports).

Dataset	N	d	C	Fold
phpAz9Len	540	39	2	1
dataset_29_credit-a	690	15	2	1
dataset_31_credit-g	1,000	20	2	2
led-display_R	1,000	7	10	2
php7KLval	1,000	20	2	3
phpMjNRWy	1,000	40	8	3
statlog-german-credit_R	1,000	24	2	4
oocytes_merluccius_nucle…	1,022	41	2	2
oocytes_merluccius_state…	1,022	25	3	2
rmftsa_sleepdata	1,024	2	2	4
qsar	1,054	41	2	1
phpGUrE90	1,055	41	2	3
solar-flare	1,066	12	3	4
phpchCuL5	1,080	81	8	3
phpmRPvKy	1,100	12	5	3
pc1	1,109	21	2	3
socmob	1,156	5	2	4
hill-valley	1,212	100	2	2
php3isjYz	1,212	100	2	3
baseball	1,340	16	3	1
php50jXam	1,372	4	2	3
pc4	1,458	37	2	3
contrac_R	1,473	9	3	1
dataset_23_cmc	1,473	9	3	2
dataset_185_yeast	1,484	8	10	2
yeast_R	1,484	8	10	4
fitness	1,500	6	2	1
pc3	1,563	37	2	3
phptd5jYj	1,593	256	10	4
semeion_R	1,593	256	10	4
wine-quality-red_R	1,599	11	6	4
Is_this_a_good_customer	1,723	13	2	1
car_R	1,728	6	4	1
php2jDIhh	1,728	6	4	3
php5s7Ep8	1,941	27	7	3
php9xWOpn	1,941	33	2	3
steel-plates_R	1,941	27	7	4
compass	2,000	17	2	1
dataset_12_mfeat-factors	2,000	216	10	1
dataset_14_mfeat-fourier	2,000	76	10	1
dataset_16_mfeat-karhune…	2,000	64	10	1
dataset_18_mfeat-morphol…	2,000	6	10	1
dataset_22_mfeat-zernike	2,000	47	10	2
mfeat-karhunen	2,000	64	10	2
mfeat-morphological	2,000	6	2	2
rl_seed	2,000	12	2	1
kc1	2,109	21	2	2
cardiotocography-10clase…	2,126	21	10	1
cardiotocography-3clases…	2,126	21	3	1
php9HX2u8	2,126	35	10	3
titanic_R	2,201	3	2	4
dataset_36_segment	2,310	19	7	2
image-segmentation	2,310	18	7	2
phpyM5ND4	2,310	19	7	4
statlog-image_R	2,310	18	7	4
hazelnut	2,400	30	2	1
phpiubDlf	2,500	100	3	3
pumpkinseed	2,500	12	2	1
phpdReP6S	2,534	72	2	3
ozone_R	2,536	72	2	3
molec-biol-splice_R	3,190	60	3	2
chess-krvkp_R	3,196	36	2	1
dataset_3_kr-vs-kp	3,196	36	2	1
abalone_R	4,177	8	3	1
bank_R	4,521	16	2	1
dataset_44_spambase	4,601	57	2	2
spambase_R	4,601	57	2	4
phpcSeK3V	4,839	5	2	3
wine-quality-white_R	4,898	11	7	4
churn	5,000	20	2	1
dataset_60_waveform-5000	5,000	40	3	2
waveform-noise_R	5,000	40	3	4
waveform_R	5,000	21	3	4
phpZrCzJR	5,100	36	2	4
php8Mz7BG	5,404	5	2	3
phpVeNa5j	5,456	24	4	4
wall-following_R	5,456	24	4	4
dataset_30_page-blocks	5,473	10	5	2
dataset_28_optdigits	5,620	64	10	2
polish_bank	5,910	64	2	1
phpPbCMyg	6,118	51	6	3
musk	6,598	169	2	2
dataset_24_mushroom	8,124	22	2	2
mushroom_R	8,124	21	2	2
phpYLeydd	9,873	32	5	4
JapaneseVowels	9,961	14	9	2

C Datasets and fold assignment

D Additional results

This appendix reports the 300-epoch counterparts of the main-text tables (Tables 8 and 10), augmentation AUC statistics (Table 11), every metric at every checkpoint for every method (Tables 12 to 15), the size-stratified results for SVM and Random Forest (Tables 16 and 17), the per-schema distribution of paired differences (Figure 3), the budget-mismatched comparison of Section 4.5 (Table 9), and the robustness checks referred to in Section 4.6 (Tables 18 and 19).

Table 8: Held-out results at 300 epochs (the longest budget we trace), averaged over 86 unseen schemas. Same conventions as Table 1.

Evaluator	Metric	Real only	CTGAN	TVAE	TabPFGen	GRAFT
XGBoost	Augmentation acc. (↑)	0.852	0.959	0.959	0.966	**0.967**
	Augmentation AUC (↑)	0.912	0.985	0.978	0.982	**0.986**
	ML-efficacy acc. (↑)	–	0.576	**0.744**	0.726	0.456
	C2ST acc. (↓)	–	0.849	**0.748**	0.815	0.985
SVM	Augmentation acc. (↑)	0.779	0.785	0.789	0.805	**0.826**
	Augmentation AUC (↑)	0.834	0.858	0.869	**0.888**	0.880
	ML-efficacy acc. (↑)	–	0.534	**0.682**	0.585	0.435
	C2ST acc. (↓)	–	0.729	0.643	**0.637**	0.948
Random Forest	Augmentation acc. (↑)	0.851	0.963	0.962	0.968	**0.970**
	Augmentation AUC (↑)	0.910	0.988	0.980	0.983	**0.989**
	ML-efficacy acc. (↑)	–	0.577	**0.764**	0.728	0.433
	C2ST acc. (↓)	–	0.810	**0.732**	0.790	0.983

Table 9: Budget-mismatched comparison: GRAFT after 5 or 10 adaptation epochs against CTGAN and TVAE after 300 training epochs, and against training-free TabPFGen. Paired augmentation-accuracy statistics across 86 schemas.

Evaluator	GRAFT epochs	vs. CTGAN@300		vs. TVAE@300		vs. TabPFGen	
		Mean Δ (win)	p	Mean Δ (win)	p	Mean Δ (win)	p
XGBoost	5	+0.009 (65%)	$<10^{-4}$	+0.007 (57%)	0.0010	+0.000 (33%)	0.355
	10	+0.011 (65%)	$<10^{-4}$	+0.008 (55%)	$<10^{-4}$	+0.001 (41%)	0.940
SVM	5	+0.039 (80%)	$<10^{-4}$	+0.027 (79%)	$<10^{-4}$	+0.014 (41%)	0.803
	10	+0.039 (83%)	$<10^{-4}$	+0.027 (81%)	$<10^{-4}$	+0.014 (45%)	0.870
Random Forest	5	+0.009 (62%)	$<10^{-4}$	+0.008 (59%)	0.0001	+0.001 (36%)	0.666
	10	+0.011 (66%)	$<10^{-4}$	+0.009 (58%)	$<10^{-4}$	+0.002 (47%)	0.257

Table 10: Paired augmentation-accuracy statistics at 300 epochs (conventions as in Table 2).

Evaluator	GRAFT vs.	Win	Tie	Loss	Mean Δ	Median Δ [IQR]	Wilcoxon p
XGBoost	CTGAN	65%	16%	19%	+0.008	+0.003 [+0.000, +0.012]	$<10^{-4}$
	TVAE	59%	17%	23%	+0.009	+0.004 [+0.000, +0.015]	$<10^{-4}$
	TabPFGen	42%	14%	44%	+0.001	+0.000 [-0.005, +0.004]	0.794
SVM	CTGAN	87%	6%	7%	+0.041	+0.025 [+0.009, +0.069]	$<10^{-4}$
	TVAE	85%	2%	13%	+0.037	+0.023 [+0.007, +0.065]	$<10^{-4}$
	TabPFGen	48%	10%	42%	+0.021	+0.000 [-0.009, +0.019]	0.120
Random Forest	CTGAN	67%	14%	19%	+0.007	+0.003 [+0.000, +0.011]	$<10^{-4}$
	TVAE	66%	12%	22%	+0.008	+0.004 [+0.000, +0.014]	$<10^{-4}$
	TabPFGen	41%	20%	40%	+0.002	+0.000 [-0.005, +0.004]	0.859

Table 11: Paired augmentation-AUC statistics at the 10-epoch budget (conventions as in Table 2).

Evaluator	GRAFT vs.	Win	Tie	Loss	Mean Δ	Median Δ [IQR]	Wilcoxon p
XGBoost	CTGAN	43%	34%	23%	+0.003	+0.000 [+0.000, +0.006]	0.008
	TVAE	52%	27%	21%	+0.008	+0.001 [+0.000, +0.010]	0.0001
	TabPFGen	51%	35%	14%	+0.006	+0.001 [+0.000, +0.009]	$<10^{-4}$
SVM	CTGAN	72%	17%	10%	+0.028	+0.012 [+0.000, +0.051]	$<10^{-4}$
	TVAE	60%	13%	27%	+0.001	+0.008 [-0.003, +0.046]	0.003
	TabPFGen	43%	10%	47%	−0.018	+0.000 [-0.009, +0.021]	0.964
Random Forest	CTGAN	38%	36%	26%	+0.002	+0.000 [-0.001, +0.003]	0.077
	TVAE	60%	31%	8%	+0.010	+0.002 [+0.000, +0.011]	$<10^{-4}$
	TabPFGen	56%	37%	7%	+0.008	+0.002 [+0.000, +0.011]	$<10^{-4}$

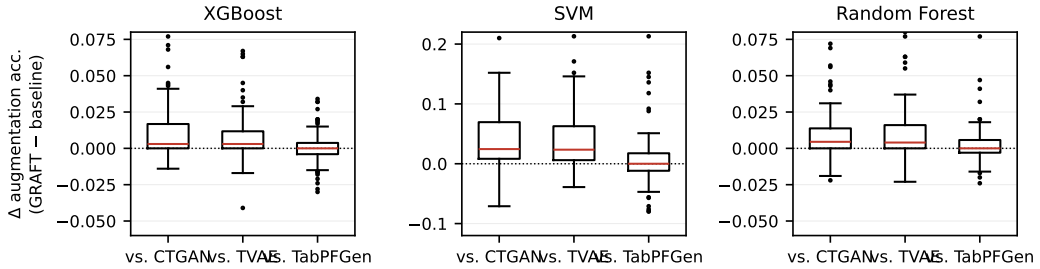

Figure 3: Per-schema paired difference in augmentation accuracy (GRAFT minus baseline) at the 10-epoch budget across the 86 held-out schemas. Boxes show the inter-quartile range, the red line the median, and dots the outliers; the dotted line marks $\Delta = 0$.

Table 12: Augmentation accuracy across adaptation/training epochs for all methods and evaluators (mean over 86 held-out schemas; TabPFGen is training-free and repeated for reference). Best per evaluator and epoch in bold.

Epochs	XGBoost				SVM				Random Forest			
	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT
1	0.955	0.956	**0.966**	0.965	0.781	0.778	0.804	**0.818**	0.959	0.959	0.968	**0.969**
5	0.958	0.958	**0.968**	0.965	0.780	0.775	0.808	**0.818**	0.961	0.960	**0.970**	0.969
10	0.957	0.959	0.965	**0.967**	0.780	0.781	0.803	**0.818**	0.961	0.961	0.967	**0.971**
20	0.956	0.957	**0.967**	0.964	0.782	0.781	0.805	**0.824**	0.961	0.960	**0.970**	0.969
50	0.958	0.959	0.966	**0.967**	0.782	0.785	0.804	**0.824**	0.961	0.961	0.969	**0.970**
100	0.957	0.958	0.966	**0.966**	0.774	0.790	0.806	**0.821**	0.961	0.961	0.968	**0.968**
200	0.956	0.959	0.966	**0.967**	0.780	0.792	0.805	**0.826**	0.960	0.962	0.968	**0.970**
300	0.959	0.959	0.966	**0.967**	0.785	0.789	0.805	**0.826**	0.963	0.962	0.968	**0.970**

Table 13: Augmentation AUC across epochs (conventions as in Table 12).

Epochs	XGBoost				SVM				Random Forest			
	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT
1	0.980	0.977	0.982	**0.986**	0.840	0.867	**0.885**	0.869	0.985	0.979	0.983	**0.989**
5	0.983	0.978	0.983	**0.986**	0.834	0.864	**0.887**	0.874	0.987	0.979	0.984	**0.990**
10	0.983	0.979	0.980	**0.986**	0.840	0.867	**0.886**	0.868	0.988	0.980	0.982	**0.990**
20	0.983	0.977	0.982	**0.986**	0.851	0.866	**0.887**	0.878	0.987	0.979	0.983	**0.990**
50	0.984	0.978	0.981	**0.987**	0.859	0.868	**0.886**	0.878	0.987	0.980	0.983	**0.991**
100	0.981	0.977	0.982	**0.987**	0.849	0.868	**0.888**	0.875	0.986	0.979	0.984	**0.990**
200	0.983	0.978	0.981	**0.987**	0.854	0.868	**0.886**	0.877	0.987	0.979	0.983	**0.991**
300	0.985	0.978	0.982	**0.986**	0.858	0.869	**0.888**	0.880	0.988	0.980	0.983	**0.989**

Table 14: ML-efficacy accuracy (train on synthetic only, test on real) across epochs; higher is better.

Epochs	XGBoost				SVM				Random Forest			
	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT
1	0.400	0.442	**0.727**	0.433	0.460	0.262	**0.583**	0.420	0.434	0.446	**0.728**	0.426
5	0.424	0.481	**0.728**	0.430	0.480	0.177	**0.587**	0.435	0.444	0.477	**0.730**	0.417
10	0.427	0.521	**0.727**	0.442	0.440	0.284	**0.585**	0.446	0.435	0.518	**0.727**	0.410
20	0.433	0.573	**0.724**	0.452	0.439	0.362	**0.586**	0.429	0.440	0.569	**0.726**	0.414
50	0.471	0.648	**0.726**	0.428	0.441	0.540	**0.588**	0.440	0.474	0.653	**0.727**	0.397
100	0.515	0.714	**0.725**	0.456	0.486	**0.623**	0.588	0.447	0.521	0.716	**0.726**	0.423
200	0.545	**0.741**	0.725	0.452	0.521	**0.673**	0.585	0.438	0.555	**0.745**	0.726	0.406
300	0.576	**0.744**	0.726	0.456	0.534	**0.682**	0.585	0.435	0.577	**0.764**	0.728	0.433

Table 15: C2ST detection accuracy across epochs; 0.5 is ideal and lower is better.

Epochs	XGBoost				SVM				Random Forest			
	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT	CTGAN	TVAE	TabPFGen	GRAFT
1	0.883	0.925	**0.816**	0.954	0.806	0.848	**0.644**	0.856	0.845	0.930	**0.790**	0.941
5	0.889	0.927	**0.816**	0.969	0.795	0.833	**0.637**	0.874	0.848	0.921	**0.788**	0.961
10	0.874	0.876	**0.814**	0.983	0.784	0.792	**0.645**	0.897	0.842	0.859	**0.790**	0.972
20	0.885	0.838	**0.820**	0.981	0.797	0.746	**0.641**	0.920	0.845	0.829	**0.789**	0.979
50	0.873	**0.803**	0.817	0.988	0.772	0.698	**0.645**	0.936	0.843	**0.785**	0.791	0.982
100	0.874	**0.777**	0.815	0.989	0.755	0.672	**0.645**	0.930	0.833	**0.760**	0.788	0.986
200	0.856	**0.758**	0.814	0.986	0.741	0.647	**0.635**	0.945	0.819	**0.741**	0.791	0.988
300	0.849	**0.748**	0.815	0.985	0.729	0.643	**0.637**	0.948	0.810	**0.732**	0.790	0.983

Table 16: Augmentation accuracy (SVM) by held-out dataset size at the 10-epoch budget (conventions as in Table 3).

Held-out size	#	Real only	CTGAN	TVAE	TabPFGen	GRAFT	Δ vs. CTGAN / TabPFGen
Smallest quintile ($N \leq 1,156$)	17	0.694	0.699	0.700	0.695	**0.741**	+0.043 / +0.046
$N \leq 1{,}000$	7	0.634	0.648	0.647	0.569	**0.686**	+0.038 / +0.117
$1{,}000 < N \leq 2{,}000$	39	0.758	0.751	0.754	0.786	**0.801**	+0.050 / +0.014
$2{,}000 < N \leq 5{,}000$	27	0.807	0.818	0.821	0.850	**0.850**	+0.032 / +0.000
$N > 5{,}000$	13	0.851	0.856	0.853	**0.883**	0.878	+0.021 / −0.005

Table 17: Augmentation accuracy (Random Forest) by held-out dataset size at the 10-epoch budget (conventions as in Table 3).

Held-out size	#	Real only	CTGAN	TVAE	TabPFGen	GRAFT	Δ vs. CTGAN / TabPFGen
Smallest quintile ($N \leq 1,156$)	17	0.796	0.932	0.933	0.943	**0.955**	+0.022 / +0.012
$N \leq 1{,}000$	7	0.725	0.908	0.910	0.920	**0.941**	+0.034 / +0.021
$1{,}000 < N \leq 2{,}000$	39	0.819	0.954	0.954	0.963	**0.965**	+0.011 / +0.003
$2{,}000 < N \leq 5{,}000$	27	0.898	0.973	0.973	0.977	**0.978**	+0.005 / +0.001
$N > 5{,}000$	13	0.929	0.984	0.985	0.987	**0.988**	+0.004 / +0.001

Table 18: Per-fold augmentation accuracy at the 10-epoch budget. Each fold holds out a disjoint set of schemas that the shared generator never saw during that round of pretraining.

Fold	#	CTGAN			TVAE			TabPFGen			GRAFT		
		XGB	SVM	RF	XGB	SVM	RF	XGB	SVM	RF	XGB	SVM	RF
1	24	0.958	0.783	0.968	0.963	0.789	0.969	0.969	0.826	0.974	0.967	0.822	0.975
2	22	0.959	0.817	0.962	0.961	0.818	0.965	0.967	0.826	0.968	0.968	0.848	0.972
3	20	0.954	0.676	0.958	0.960	0.671	0.958	0.962	0.711	0.962	0.967	0.738	0.969
4	20	0.955	0.839	0.954	0.951	0.841	0.951	0.962	0.843	0.964	0.965	0.862	0.966

Table 19: Stability across the eight checkpoints. Left: standard deviation of the mean augmentation accuracy across epochs. Right: per-schema standard deviation of augmentation accuracy across epochs, averaged over schemas (lower means a method delivers the same augmentation quality regardless of budget).

Evaluator	**Std. of mean across epochs**			**Mean per-schema std. across epochs**		
	CTGAN	TVAE	GRAFT	CTGAN	TVAE	GRAFT
XGBoost	0.0013	**0.0010**	0.0012	0.0083	0.0081	**0.0066**
SVM	**0.0033**	0.0060	0.0035	0.0190	0.0170	**0.0122**
Random Forest	0.0013	0.0010	**0.0008**	0.0083	0.0072	**0.0060**