A NATIVELY DIFFERENTIALLY PRIVATE FOUNDATION MODEL FROM NOISY CHEBYSHEV MOMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models such as TabPFN predict by in-context learning: the training rows of a table are placed in the model context, and one forward pass returns predictions for the test rows. The scheme is incompatible with differential privacy (DP), because every prediction is then a deterministic function of the sensitive rows. To address this gap, we present NoisyMomentPFN, a tabular foundation model designed for inputs that are already a differentially private object. The data holder publishes, once, the Chebyshev moments of the target along public random projections of the features, together with the Gram matrix of the columns, noised in a single Gaussian mechanism. A prior-fitted transformer is pretrained on millions of synthetic and public tables *whose releases are noised on the fly*, so that it learns a prior over noisy statistical moment releases; at test time it reads the DP moment release of a new table and returns a predictive distribution for any query. All queries are then post-processing, and incur no additional privacy budget. On a large suite of tabular datasets, NoisyMomentPFN beats the strongest one-shot baseline, sufficient statistics perturbation for ridge regression, at every privacy budget in aggregate, and at moderate budgets returns better predictive distributions, as measured by CRPS, than a neural network trained per table with DP-SGD, orders of magnitude faster.

1 INTRODUCTION

History will likely remember the 2020s as a boom time for foundation models. Nearly every prediction setting now has a general-purpose model driving state-of-the-art performance: text, images, and, most recently, tabular data (Bommasani et al., 2021; Hollmann et al., 2025). Tabular foundation models such as TabPFN and TabFM (Hollmann et al., 2025; Google Research, 2026) are pretrained once on a massive *synthetic* prior of tables, and at inference time are shown a target table through *in-context learning* (ICL).[1] Concretely, the training rows of a new table are supplied as a context C, and a single forward pass returns $q_\phi(\cdot \mid C, X)$, a distribution over the targets of the test rows X; in this process the pretrained parameters ϕ remain frozen, and there is *no training* on the new table. Perhaps surprisingly, the best of these models now outperform gradient-boosted trees that are *trained* on the data (Hollmann et al., 2025; Grinsztajn et al., 2025), with no per-table optimization at all: "training" amounts to placing the rows in context, and prediction is a single forward pass.

Of course, this scheme is fundamentally at odds with differential privacy (DP).

Feeding *potentially sensitive* rows into the model context makes every prediction a deterministic function of those rows, so the inference scheme cannot be differentially private, and recent work even shows that membership inference against tabular ICL succeeds in practice (Wahdany et al., 2026; Capano & Böhler, 2026). There is also no obvious place to introduce noise: for a $n \times d$ table, perturbing each row before it enters the context fails because a row's sensitivity is the row itself, so each row must receive noise of order $\sqrt{d}/\epsilon$, comparable to the data and independent of n, thus a row-wise scheme preserves privacy only by destroying the signal. We are left asking: *how would a tabular foundation model be built if the training table is sensitive and requires a differentially private treatment?*

[1] Here ICL means directly predicting the label from the supplied context, not autoregressive generation.

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.

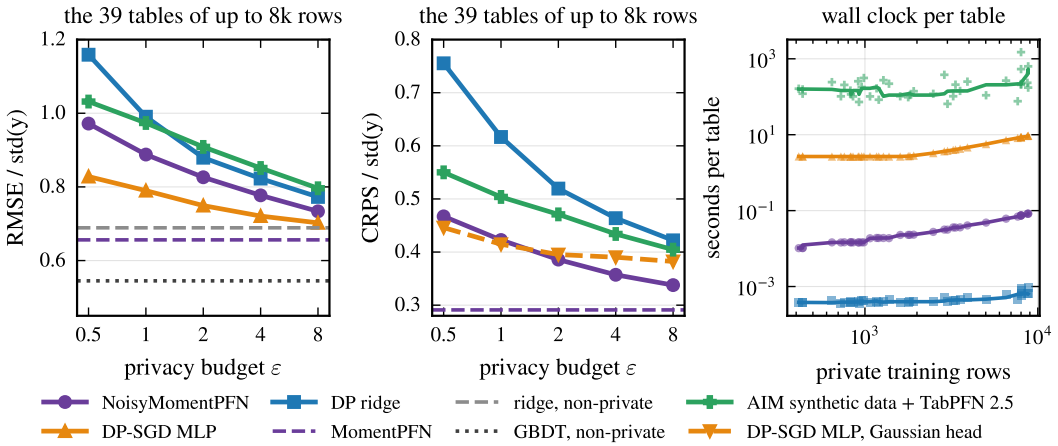


Figure 1: NoisyMomentPFN against the private baselines and the train-on-synthetic AIM pipeline. **Left**: normalized RMSE against the privacy budget. **Middle**: normalized CRPS of the predictive distribution (lower is better for both); dashed and dotted lines are non-private references, the dashed purple line is MomentPFN trained and scored *without* noise (so is the "floor" for performance of NoisyMomentPFN), and the green line is an AIM synthetic table at the same budget read in context by TabPFN-2.5 and scored on the real test rows. **Right**: measured wall clock per table. On these tables, NoisyMomentPFN improves on DP ridge at every budget, has the best CRPS from $\epsilon = 2$ on, is more accurate than the pipeline at every budget, and is two orders of magnitude faster than the DP-SGD MLP and four orders faster than the AIM pipeline.

Two natural first attempts. A natural first attempt is to simply take a pretrained prior-fitted network (e.g. TabPFN) and fine-tune it on the private table with DP-SGD (Abadi et al., 2016), which discards exactly what makes tabular foundation models appealing: their flexibility and speed without any data-specific optimization. Alternatively, one could generate a DP synthetic table from the training rows (e.g., with a state-of-the-art DP synthetic data method (McKenna et al., 2022; Zhang et al., 2017; Liu et al., 2023)) and place *that* table in the context of a non-private foundation model. This is DP by post-processing, but synthetic data generators are expensive, must be run per table, and are designed to preserve a workload of low-order marginals rather than the conditional distribution of a particular target, so the downstream accuracy is limited by how well a generic synthetic table happens to encode it (Ponomareva et al., 2026) (note we measure this gap directly in Section 4.1).

Our alternative: a natively private foundation model. We present a third option: a tabular foundation model whose *input* is already a differentially private object, so that predictions are private by post-processing. But what should this object be, if not somehow noised versions of the rows? Standard DP statistics such as low-order marginals are an awkward fit: their number and layout depend on the schema of each table (there are $\binom{d}{m}$ m-way marginals, each with its own bins), whereas a foundation model needs a vocabulary of fixed layout. Our approach instead leverages recent work on DP synthetic data via *Chebyshev moment matching* (Musco et al., 2025), which releases noisy Chebyshev polynomial moments, a basis for which recovery in Wasserstein-1 distance is near-optimally robust to noise in one dimension.

Tabular data are high-dimensional, so we project the data onto public random directions and compute Chebyshev moments along each projection, jointly with the target; we call these *sliced Chebyshev moments*, and our final model releases them for pairs of projections. Each slice is a noise-robust view of the joint distribution; every entry is an average over rows, so its sensitivity is of order $1/n$; and the number of released moments is the product of the number of direction pairs and the number of moments per pair, so it does not grow with the number of columns. The only part of the release that does is the Gram block, of size $O(d^2)$. Fixing the directions in advance gives a fixed vocabulary, and we pretrain a prior-fitted network whose context is not a table but *a noisy release*, simulating the privacy noise during pretraining so that the network learns the observation model it will meet at inference. At inference, the data holder computes the privatized statistics once, which is cheap, and the pretrained model predicts from them in a single forward pass. This leads us to our moment-based foundation model MomentPFN, and the variant NoisyMomentPFN, which is the *same* model but reading a *noisy* private release during pre-training. We clarify our contributions as follows.

Contribution 1: A natively private tabular foundation model. We propose MomentPFN, a prior-fitted transformer that reads, in place of rows, a fixed-size release of a table, composed of noisy statistics. These statistics are the sliced Chebyshev moments of the target along L public pairs of random directions, together with the row-averaged Gram matrix, the sufficient statistic of ridge regression. NoisyMomentPFN is the same model pretrained on noisy statistics (using the Gaussian mechanism). In both cases, the pretraining objective is minimized by the Bayes-optimal predictor for a task from the prior seen only through the release of the (potentially noisy) statistics (Section 3.3).

Contribution 2: State-of-the-art one-shot private prediction. On 52 real regression tables and 33 classification tables, at $\epsilon \in \{0.5, 1, 2, 4, 8\}$ with $\delta = n^{-2}$, NoisyMomentPFN improves on the strongest one-shot baseline, sufficient statistics perturbation for ridge regression (Wang, 2018), at every budget in aggregate and on every stratum of tables under 10k rows. It returns better predictive distributions than training a tabular MLP (Gorishniy et al., 2021) with DP-SGD, with the best CRPS of any private method from $\epsilon = 2$ on. Additionally, NoisyMomentPFN is more accurate and faster than a pipeline that hands a state-of-the-art DP synthetic table, generated by AIM, to a non-private tabular foundation model, e.g., TabPFN v2.5. The DP-SGD MLP does achieve lower RMSE on most tables, but at a different cost: NoisyMomentPFN answers every query from one release in tens of milliseconds, whereas the MLP is trained anew for every table and budget, at hundreds of times the wall clock on the same GPU.

2 PRELIMINARIES

We recall the two most important ingredients our method builds on below. A detailed discussion of related work is deferred to Section A.

Differential privacy. A tabular dataset S consists of n rows (x_i, y_i), rescaled with public per-column bounds so that $x_i \in [-1, 1]^d$ and $y_i \in [-1, 1]$ for regression (or $y_i \in \{1, \dots, C\}$ for classification). Two tables of the same size are *neighboring*, written $S \sim S'$, if they differ in one row. A randomized mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private if, for all neighboring S, S' and every measurable set $\mathcal{B}$ of outputs, $\Pr[\mathcal{M}(S) \in \mathcal{B}] \leq e^{\epsilon} \Pr[\mathcal{M}(S') \in \mathcal{B}] + \delta$ (Dwork & Roth, 2014). We use a single mechanism throughout, the analytic Gaussian mechanism: for a vector-valued query v with ℓ_2 sensitivity $\Delta = \max_{S \sim S'} \|v(S) - v(S')\|_2$, releasing $v(S) + \sigma\xi$ with $\xi \sim N(0, I)$ is (ϵ, δ)-DP for every $\epsilon > 0$ whenever $\sigma \geq \sigma_{\mathrm{AG}}(\Delta, \epsilon, \delta)$, the analytic calibration of Balle & Wang (2018). Any data-independent function of a DP release is *post-processing* and incurs no further privacy cost.

(Sliced) Chebyshev moments. Let T_j be the jth Chebyshev polynomial of the first kind, defined as $T_0(x) = 1$, $T_1(x) = x$, and $T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x)$. The *normalized* polynomials $\bar{T}_0 = 1/\sqrt{\pi}$ and $\bar{T}_j = \sqrt{2/\pi}T_j$ for $j \geq 1$ are orthonormal under the weight $(1 - x^2)^{-1/2}$ on $[-1, 1]$ and bounded by $\left|\bar{T}_j(x)\right| \leq \gamma_j$, with $\gamma_0 = 1/\sqrt{\pi}$ and $\gamma_j = \sqrt{2/\pi}$ for $j \geq 1$. The Chebyshev moments $\mathbb{E}_{x \sim p}[T_j(x)]$ of a distribution p on $[-1, 1]$ are far better conditioned than monomial moments for recovering p from noisy measurements. In particular, the jth moment tolerates noise of order $\sqrt{j}$ times the noise on the first without spoiling $O(1/k)$ recovery in Wasserstein-1 distance from the first k moments (Musco et al., 2025). This robustness is a one-dimensional statement, and it extends to a distribution p on the unit ball of $\mathbb{R}^d$ through its projections. For a unit direction θ, the *sliced* Chebyshev moments of p are the Chebyshev moments of $\langle \theta, x \rangle \in [-1, 1]$, namely $\mathbb{E}_{x \sim p}[T_j(\langle \theta, x \rangle)]$.

3 METHODOLOGY

We first describe how the statistics are computed from the training rows and released privately, then specify the predictor and how it uses the release to make predictions, and finally give the pretraining and inference procedures. We present the regression setting with the two-dimensional pair release used in our experiments. The one-dimensional release and the classification variant are deferred to Section B. Throughout, a *task* is a distribution P over labeled pairs (X, Y) with $X \in [-1, 1]^d$ and $Y \in [-1, 1]$,[2] tasks are drawn from a meta-distribution Π, and a table S of n rows is drawn i.i.d.

[2]Raw data are brought into this range by an affine rescaling with public per-column bounds, then clipped to $[-1, 1]$; categorical columns are first mapped to integer codes, and missing cells map to the midpoint 0. We assume the bounds are part of a public schema (Section D).

from P. We write $\mathbf{p}$ for everything a release depends on other than the rows of S: the sizes n and d, the rescaling bounds, the budget (ϵ, δ), the Chebyshev degree k, the L public pairs of unit directions $(\theta_\ell, \theta'_\ell)$, the projection scale $s > 0$, and the Gram weight $w > 0$ introduced below. At inference, all of $\mathbf{p}$ is fixed without looking at S.

3.1 THE PRIVATE RELEASE

Statistics. Half of the L pairs are coordinate-axis pairs (e_i, e_j), ordered pairs with $i \neq j$ drawn uniformly with replacement, so that interactions between two columns are read without a projection, and the rest are pairs of independent uniform directions on the sphere. For row i and pair ℓ, let $u_{i\ell} = \mathrm{clip}(\langle X_i, \theta_\ell \rangle / s, -1, 1)$ and $u'_{i\ell} = \mathrm{clip}(\langle X_i, \theta'_\ell \rangle / s, -1, 1)$ be the scaled, clipped projections onto the two directions of the pair. he release consists of two kinds of statistics, both formed by averaging bounded quantities over the rows: the *moment block* of pair ℓ and the *Gram block*

$$M_\ell[a,b,c] = \frac{1}{n}\sum_{i=1}^{n} \bar{T}_a(u_{i\ell})\bar{T}_b(u'_{i\ell})\bar{T}_c(Y_i), \quad a,b,c \in \{0,\dots,k\}, \qquad G = \frac{1}{n}\sum_{i=1}^{n} z_i z_i^\top, \tag{1}$$

where $z_i = (X_i, 1, Y_i) \in \mathbb{R}^{d+2}$. We note that M_ℓ is the $(k+1)^3$ tensor of Chebyshev moments of the joint distribution of the two projections and the target along pair ℓ, the coefficients of a truncated expansion of their joint density in the product basis, and G holds the sufficient statistics of ridge regression of Y on $(X, 1)$ (Wang, 2018). We drop the constant entry $M_\ell[0,0,0] = \pi^{-3/2}$ and keep the upper triangle of G (its constant entry $G_{d+1,d+1} = 1$ included), so the release $R(S) = (M_1, \dots, M_L, G)$ has $L[(k+1)^3 - 1] + N_G$ scalars with $N_G = (d+2)(d+3)/2$, independent of n.

Weighting and sensitivity. Following the one-dimensional mechanism of Musco et al. (2025), in which the jth moment tolerates noise proportional to $\sqrt{j}$, we allocate less noise to low orders. Let $s_{abc} = \sqrt{\max(a,1)\max(b,1)\max(c,1)}$ and form the vector

$$v(S) = \Big(\tfrac{1}{s_{abc}} M_\ell[a,b,c]\Big)_{\ell \in [L],\ (a,b,c) \neq \mathbf{0}} \ \oplus\ \big(w G_{jj'}\big)_{j \leq j'}, \tag{2}$$

where $w > 0$ trades noise between the two blocks: after the weights are undone, the Gram block carries $1/w$ of the noise of the moment entries with $s_{abc} = 1$. Every entry of $v(S)$ is an average over rows of a function bounded in absolute value, by β_{abc}/s_{abc} with $\beta_{abc} = \gamma_a\gamma_b\gamma_c$ for the moment entries and by w for the Gram entries, so replacing one row moves it by at most $2/n$ times that bound, and for neighbors $S \sim S'$,

$$\|v(S) - v(S')\|_2^2 \leq \frac{4}{n^2}\Big(L \sum_{(a,b,c)\neq\mathbf{0}} \frac{\beta_{abc}^2}{s_{abc}^2} + w^2 N_G\Big) =: \Delta^2, \qquad \sum_{(a,b,c)\neq\mathbf{0}} \frac{\beta_{abc}^2}{s_{abc}^2} = \Big(\frac{1+2H_k}{\pi}\Big)^3 - \frac{1}{\pi^3}, \tag{3}$$

with H_k the kth harmonic number. We emphasize that, as in Musco et al. (2025),the moment part of the sensitivity grows only polylogarithmically in the degree.

Mechanism. The mechanism releases $v(S) + \sigma\xi$ with $\sigma = \sigma_{\mathrm{AG}}(\Delta, \epsilon, \delta)$ and $\xi \sim \mathcal{N}(0, I)$. Multiplying the moment coordinates back by s_{abc} and dividing the Gram coordinates by w, a fixed linear map of the noised vector, gives the released blocks

$$\tilde{M}_\ell[a,b,c] = M_\ell[a,b,c] + s_{abc}\sigma\xi_{\ell abc}, \qquad \tilde{G} = G + \frac{\sigma}{w}\Omega, \tag{4}$$

where Ω is the symmetric matrix whose upper triangle holds the Gram coordinates of ξ, and the constant entry $\tilde{M}_\ell[0,0,0] = \pi^{-3/2}$, which was not released, is filled back in.. We write $\tilde{R} = \mathcal{M}_{\mathbf{p}}(S; \xi)$ for this release (Algorithm 1 in Section C). Note that equation 4 is not a second mechanism, but rather only rewrites the noised vector on the original blocks.

Proposition 1 (Privacy of the release)**.** *Fix public parameters $\mathbf{p}$ chosen without looking at S. The randomized mechanism $S \mapsto \mathcal{M}_{\mathbf{p}}(S; \xi)$ with $\xi \sim \mathcal{N}(0, I)$ is (ϵ, δ)-differentially private with respect to the replace-one relation of Section 2, tables of size n differing in one row. Consequently every quantity computed from $\tilde{R}$ and $\mathbf{p}$ without further access to S, including every prediction of the model of Section 3.2 at every query point (neither ϕ nor the query points depend on S), is (ϵ, δ)-differentially private.*

Proof. By Equation 3, v has ℓ_2 sensitivity at most Δ over neighboring tables of size n, using only that each row lies in $[-1,1]^d \times [-1,1]$ after the public rescaling and that the projections are clipped to $[-1,1]$. Adding $\mathcal{N}(0,\sigma^2 I)$ with $\sigma = \sigma_{\mathrm{AG}}(\Delta,\epsilon,\delta)$ is (ϵ,δ)-DP by Balle & Wang (2018); undoing the weights, filling in the constant entry, and evaluating the model are post-processing, and the two blocks are released by one mechanism applied to one vector, so there is no composition. □

3.2 THE PREDICTOR

The predictor is a network with parameters ϕ that maps the release $\tilde{R}$, the public parameters $\mathbf{p}$, and a query $x \in [-1,1]^d$ (namely, the feature vector of a test row, which need not be protected), to a distribution $q_\phi(\cdot \mid x, \tilde{R}, \mathbf{p})$ over the target of that row. For regression in particular, the distribution is a histogram over B equal bins of $[-1,1]$. The training data enters only through $\tilde{R}$, i.e. the private release of the training rows, which serves every query. The layout mirrors a prior-fitted network such as TabPFN (Hollmann et al., 2025), with the in-context training rows replaced by one token per direction pair, all of which the test row attends to.

Encoder. The encoder receives $L+1$ tokens: (i) one per pair (L in total) formed from $\tilde{M}_\ell$, its two directions, $\log\sigma$ and $\mathbf{p}$, and (ii) one Gram token, formed from the released upper triangle of $\tilde{G}$, $\log(\sigma/w)$ and $\mathbf{p}$. It then runs eight layers of self-attention over them, once per release.

Decoder. For a query x, the query token is the linear embedding of x, $\mathbf{p}$, and the ridge prediction $\hat{y}(x)$ read off the noisy Gram block with penalty $\lambda = \sqrt{d+1}\sigma/w$, which grows with the noise, together with its residual scale. The keys and values are the $L+1$ encoded tokens, the Gram token included, where each encoded pair token is summed with the embedding of its hand-crafted feature vector computed from the pair's block $\tilde{M}_\ell$ and the query's two projections $u_\ell(x) = \mathrm{clip}(\langle x,\theta_\ell\rangle/s,-1,1)$ and $u'_\ell(x) = \mathrm{clip}(\langle x,\theta'_\ell\rangle/s,-1,1)$. These feature vectors consist of (i) the bases $\bar{T}_a(u_\ell(x))$ and $\bar{T}_b(u'_\ell(x))$ for $a,b \le k$, (ii) the marginal estimate $\sum_{a,b}\tilde{M}_\ell[a,b,0]\bar{T}_a(u_\ell(x))\bar{T}_b(u'_\ell(x))$, and (iii) the truncated series

$$\hat{f}_\ell(x,y) = \sum_{a,b,c=0}^{k} \tilde{M}_\ell[a,b,c]\bar{T}_a(u_\ell(x))\bar{T}_b(u'_\ell(x))\bar{T}_c(y) \tag{5}$$

on a fixed grid of $y \in [-1,1]$, which, up to the Chebyshev weight and normalization over y, is the raw moment-matching estimate of the conditional distribution of the target given the two projections. The decoder then runs four layers of cross-attention from the query token to these keys and values, followed by an output head giving the B bin logits. Thus the decoder receives both the ridge estimate from the Gram block and a direct moment-based estimate from each pair block.

3.3 PRETRAINING AND INFERENCE

Pretraining. The generation of the pretraining data can be described abstractly as follows. For each task P_i drawn from the meta-distribution Π, we draw n_i i.i.d. training rows $S_i \sim P_i^{n_i}$ and, independently of them, K i.i.d. test rows $(X_{ij},Y_{ij})_{j\in[K]} \sim P_i^K$, i.e., the query-label pairs. As described above, the model is pretrained on releases of the training rows rather than the rows themselves, so for each task we also draw public parameters $\mathbf{p}_i \sim \mathcal{P}$ and form $\tilde{R}_i = \mathcal{M}_{\mathbf{p}_i}(S_i;\xi_i)$. The model q_ϕ[3] is then optimized on

$$\mathcal{L}(\phi) = \mathbb{E}\Big[-\frac{1}{K}\sum_{j=1}^{K}\log q_\phi\big(Y_{ij} \mid X_{ij},\tilde{R}_i,\mathbf{p}_i\big)\Big], \tag{6}$$

where the expectation is over P_i, S_i, the query rows, $\mathbf{p}_i$ and ξ_i. We pretrain two models that differ in the releases they read: MomentPFN is trained on clean releases ($\xi_i = 0$) and NoisyMomentPFN, initialized from it as described below, on releases with fresh standard Gaussian noise ξ_i, matching the private release used at DP inference. We emphasize that no privacy budget is spent in pretraining either model, since every S_i is synthetic or drawn from heldout public data. The noise ξ_i of NoisyMomentPFN serves utility only, so that the model reads noisy statistics in pretraining exactly as it will at inference.

[3]For regression, $\log q_\phi(Y \mid \cdot)$ is the log-probability of the bin containing Y.

Concretely, Π is a mixture of synthetic tables (three quarters) and real tables (one quarter) subsampled from 36 public regression tables **disjoint from every evaluation table**. Of the synthetic tables, half come from a structural causal prior in the style of TabPFN and half from the feature generators of the TabPFN prior (Hollmann et al., 2023) paired with tree, interaction and linear targets, in both cases with augmentations that reproduce the column and target types of public tables. The public parameters are drawn as $n_i \sim \mathrm{LogUniform}[10^2, 10^4]$, $d_i \sim \mathrm{Uniform}\{2, \ldots, 20\}$, $\epsilon_i \sim \mathrm{LogUniform}[0.1, 10]$, and $\delta_i = n_i^{-2}$, with fresh random direction pairs for every table and rescaling bounds drawn from several regimes;[4] the remaining constants are fixed at $L = 128$, $k = 6$, $w = 5$, $s = 1$, $B = 100$ and $K = 64$. Both models have the same architecture with 21.7M parameters. MomentPFN is trained for 60k steps on clean releases, and NoisyMomentPFN starts from its weights and is trained for a further 60k steps on noisy releases, with ϵ_i, ξ_i, and a random subset of between $L/2$ and L of the pairs redrawn each time a table enters a batch (Section B).

Inference. Given a sensitive table with training rows S_{train} and a budget (ϵ, δ), the data holder computes the private release $\tilde{R} = \mathcal{M}_{\mathbf{p}}(S_{\mathrm{train}}; \xi)$ once,[5] and discards S_{train}. The frozen pretrained NoisyMomentPFN then returns $q_\phi(\cdot \mid x, \tilde{R}, \mathbf{p})$ for every x among the test rows in a single forward pass. By Proposition 1, the entire set of predictions is (ϵ, δ)-DP with respect to the rows of S_{train}.

4 Experiments

Data. We evaluate on 52 real regression tables with at most 20 columns. The `numeric` stratum (17 tables) and the `mixed` stratum (22 tables, containing categorical columns) hold tables of at most $10,000$ rows, the fixed-split regression tables of TALENT (Liu et al., 2025) together with seven tables from OpenML-CTR23 (Fischer et al., 2023), and the `large` stratum (13 tables) holds tables from OpenML benchmarks (Grinsztajn et al., 2022; Erickson et al., 2025; Gijsbers et al., 2024) with the training set truncated to $50,000$ rows. Section E lists every table.

Methods. We compare NoisyMomentPFN, pretrained as in Section 3.3, against two private baselines on every table and budget $\epsilon \in \{0.5, 1, 2, 4, 8\}$ with $\delta = n^{-2}$. ***DP ridge*** is sufficient statistics perturbation (Wang, 2018): one analytic Gaussian release of the Gram matrix of $(X, 1, Y)$ under the same replace-one neighboring relation as our release, then ridge regression on it with strength $\sqrt{d+1}\sigma$, the shape of the AdaSSP rule without its data-dependent term, which was the strongest of the fast one-shot baselines we tried. The ***DP-SGD MLP*** is based on the three-layer network from Gorishniy et al. (2021)[6] trained from scratch on each table using Opacus (Yousefpour et al., 2021), with per-example clipping norm 1, Poisson subsampling for 50 epochs, and a noise multiplier calibrated by Renyi accountant (Mironov, 2017). See Section F for details.

We also consider the paradigm in which a non-private foundation model reads a private synthetic copy of the training rows generated by AIM (McKenna et al., 2022), a state-of-the-art marginal-based generator, at the same budget. As non-private references we report ridge regression, gradient boosting, and, for runtime, a public MLP with the architecture of the DP-SGD MLP and two tabular foundation models, TabFM-1.0 (Google Research, 2026) and TabPFN-2.5 (Grinsztajn et al., 2025).

Metrics. We measure point accuracy by the RMSE of the predictive mean and distributional accuracy by the continuous ranked probability score (CRPS) of the predictive distribution (Gneiting & Raftery, 2007), each divided by the standard deviation of the test target and averaged over tables with equal weight, and we also report the empirical coverage of the central 90% interval. The CRPS is the squared distance between the predictive cumulative distribution and the step function at the realized target, integrated over the target range; it reduces to the absolute error for a point prediction and rewards a distribution that is both sharp and well placed. Cost is measured by the wall clock per table on one GPU,[7] together with the peak GPU memory.

[4] Including an empirical range and clipped percentiles; the model should not assume data "fills its box."

[5] The Chebyshev bases of the projections and the target is a single matrix product on a GPU.

[6] For RMSE we use a point-prediction head trained with the squared loss; for CRPS and coverage, a Gaussian head that outputs a mean and a log-variance and is trained with the Gaussian negative log-likelihood.

[7] We measure the release plus prediction time for NoisyMomentPFN and DP ridge, and the training plus prediction time for the DP-SGD MLP, where the release of NoisyMomentPFN is computed on the GPU and that of DP ridge on the CPU.

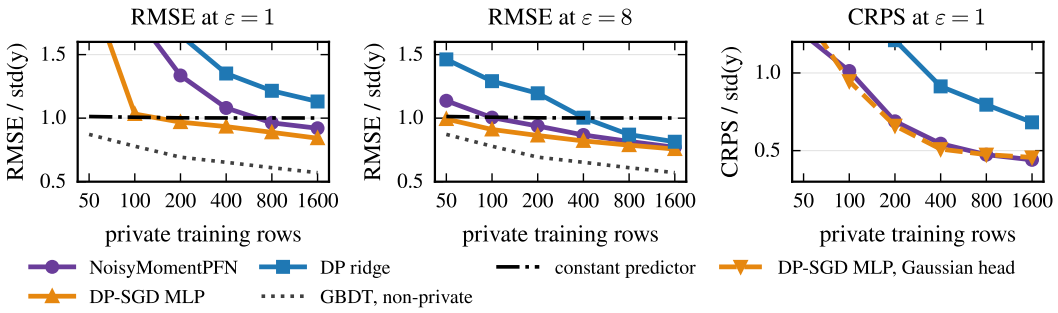


Figure 2: Utility against the number of private training rows, pooled over the 21 tables with at least $1,600$ rows, each subsampled to the sizes shown with a fixed test split and $\delta = 1/m^2$ for m rows. Left and middle: normalized RMSE at $\epsilon = 1$ and $\epsilon = 8$ against the constant predictor and non-private boosting; right: normalized CRPS at $\epsilon = 1$ with the Gaussian-head DP-SGD MLP. Lines that leave the top of a panel are worse than the range shown.

4.1 Main Results

Utility. Figure 1 compares NoisyMomentPFN with the private baselines and the non-private references over 39 tables (the tables that we were able to run the full AIM pipeline on). Section G.1 reports the aggregate over all 52 tables, excluding AIM, and the coverage of the 90% interval (Figures 5 and 6) and splits every metric by stratum (Figures 7 to 9); the picture is the same there. On RMSE, over these 39 tables NoisyMomentPFN improves over DP ridge at every budget, 0.734 against 0.773 at $\epsilon = 8$, winning 24 to 31 of the 39 tables, with the largest gain at small budgets, coming within even 0.05 of non-private ridge (0.689) at $\epsilon = 8$. The DP-SGD MLP attains the lowest RMSE at every budget (0.828 to 0.702), at the cost of a private training run for *every* table and budget. On CRPS, however, NoisyMomentPFN is level with the DP-SGD MLP at $\epsilon = 1$ (0.423 against 0.415) and the *best method from* $\epsilon = 2$ *on*: the DP-SGD MLP plateaus at 0.38 to 0.40 from $\epsilon = 2$, whereas NoisyMomentPFN keeps improving to 0.338 at $\epsilon = 8$, ahead on 25 of 39 tables, and DP ridge is far behind at every budget. Coverage tells the same story (Figure 6): NoisyMomentPFN covers 0.72 to 0.74 at every budget, whereas the coverage of both baselines drifts with the budget. The per-stratum picture (Section G.1) is the same, with the RMSE margin to the DP-SGD MLP largest on the numeric tables (0.13 at $\epsilon = 1$) and smallest on the large tables (0.05).

Figure 1 also shows MomentPFN trained on clean statistics and evaluated on clean statistics, which is not private, in order to separate two sources of error: the cost of the privacy noise and the cost of the representation itself. The gap between NoisyMomentPFN and MomentPFN shrinks quickly with the budget. Specifically, MomentPFN reaches an RMSE of 0.656, below non-private ridge at 0.689, showing that the moment release contains nonlinear predictive information beyond the Gram matrix. It remains above non-private boosting at 0.545, which indicates that the representation of the release itself still limits performance. We study this gap further in Section G.5.

A closer look at utility on tables with few rows. At first glance the RMSE gap between the DP-SGD MLP and NoisyMomentPFN at small budgets is large, but the aggregate hides a dependence on table size. Shown in Figure 2, the number of rows needed to beat the constant predictor gives a rough measure of sample efficiency: at $\epsilon = 1$ the DP-SGD MLP needs between 100 and 200 rows, NoisyMomentPFN between 400 and 800, and DP ridge more than $1,600$; at $\epsilon = 8$ the thresholds drop to below 100, between 100 and 200, and between 400 and 800 rows. The ordering is the same but the differences shrink with the budget, and at $\epsilon = 8$ NoisyMomentPFN and DP-SGD MLP are within 0.08 of each other from 200 rows on and within 0.015 at $1,600$ rows. The CRPS gap closes earlier. At $\epsilon = 1$, NoisyMomentPFN competes with DP-SGD MLP from around 200 rows onwards.

Below a few hundred rows, we hypothesize that the main failure mode of NoisyMomentPFN is a shifted predictive mean. The release spreads the privacy budget across thousands of moment coordinates, so at 100 rows even the target mean is difficult to recover accurately. In contrast, the bias gradient of DP-SGD in DP-SGD MLP training is consistently signed and the network learns the mean almost for free. Thus NoisyMomentPFN needs more rows before its point predictions become competitive, but the gap closes steadily with either more rows or a larger privacy budget, and its distributional predictions become competitive substantially earlier.

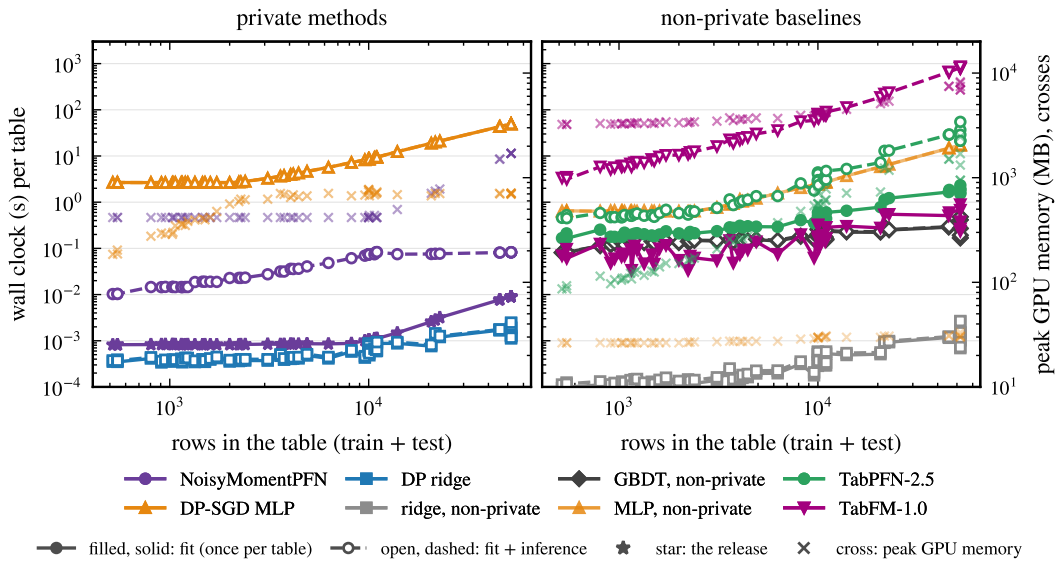


Figure 3: Cost per table against the number of rows, on one GPU, one point per table (the median over budgets and draws). Filled markers with solid lines: fitting once per table (for NoisyMomentPFN, the star is the release alone); open markers with dashed lines: fitting plus inference on all test rows; crosses: peak GPU memory (right axis; CPU methods have none). Left: the private methods; right: the non-private references.

Computational cost. Figure 3 reports the wall clock per table and the peak GPU memory of every method against the number of rows. The release of NoisyMomentPFN is one matrix product on the GPU, and it costs under a millisecond on the median table and 9 ms at 50k rows, the same order as DP ridge. For example, the full inference pipeline takes 15 ms on tables of under $2,000$ rows, 36 ms on the median table, and 82 ms at 50k rows. In contrast, DP-SGD MLP takes 2.7 s on small tables and 49 s at 50k rows, $141\times$ longer than NoisyMomentPFN on the median table (110 to 605 times across tables), and the non-private MLP and the foundation models reading the raw rows are in the same range or slower, where TabPFN-2.5 takes 1.1 s on the median table and 29 s at 50k rows, TabFM-1.0 23 s and 13 minutes. A bigger difference is how cost scales with the number of rows. The NoisyMomentPFN release has a fixed size, so its inference time is nearly flat above 10k rows, whereas every other network-based method grows steadily with the number of rows. Memory follows the same pattern, with NoisyMomentPFN below 1.7 GB against up to 8 GB for TabFM-1.0.

Foundation models on private synthetic data. We next compare against the paradigm in which TabPFN-2.5 reads a private synthetic table generated by AIM at each privacy budget, on the 39 tables of the numeric and mixed strata. Figure 1 shows that NoisyMomentPFN is more accurate at

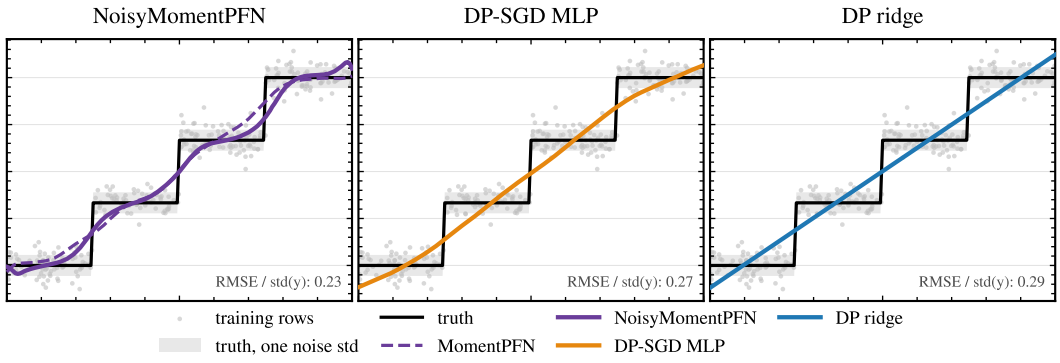


Figure 4: The predictive mean of each private method (one panel each) against a known function (staircase) and a subsample of the training rows (drawn noisily from the function, at $\epsilon = 1$ with $10,000$ rows. The dashed purple line is MomentPFN without noise, and the corner of each panel gives the method's normalized test RMSE.

every budget on both metrics: on RMSE 0.888 against 0.974 at $\epsilon = 1$ and 0.734 against 0.796 at $\epsilon = 8$, where the synthetic pipeline only reaches the level of DP ridge, and on CRPS 0.423 against 0.504 and 0.338 against 0.405. This observation is consistent with the mechanism differences of the two methods. AIM is designed to preserve low-order marginals, rather than the conditional structure of the target that the predictor needs, while NoisyMomentPFN releases statistics that directly couple the features with the target. The AIM pipeline is also by far the most expensive method in the comparison (right panel of Figure 1): generating one synthetic table takes AIM 200 s on the median table and up to nine hours on the widest ones, against 23 ms for NoisyMomentPFN on the median of the same tables (the 36 ms above includes the large tables).

A one-dimensional illustration. Figure 4 intuitively demonstrates what the predictive mean of NoisyMomentPFN looks like: a "staircase" like function with four steps with homoscedastic noise, $10,000$ training rows at $\epsilon = 1$. The release describes the conditional distribution of the target along each projection at a resolution set by the Chebyshev degree, so NoisyMomentPFN recovers the steps, with rounded corners where the noise-free MomentPFN is sharper, at a normalized test RMSE of 0.24; the DP-SGD MLP smooths the steps into a ramp (0.25), and DP ridge can only fit a line (0.29). In Section G.7, we show the experiment for five more functions at $\epsilon = 1$.

4.2 Further Results

We summarize further results whose details are deferred to the appendix.

Classification. The classification model releases, per direction, the Chebyshev moments of the projections of each class rather than the target moments (Section B). Across the 33 classification tables (Section G.2, Figure 10) the ordering mirrors regression: NoisyMomentPFN beats the DP one-vs-rest ridge at every budget (0.652 against 0.638 mean accuracy at $\epsilon = 1$) and the DP-SGD MLP remains ahead by 0.01 to 0.03 across budgets.

Additional analyses and ablations. Section G contains detailed results not shown in the main figures. We additionally evaluate these methods on the large-table suite at both 10k and 50k rows in Section G.3, where we expect that a reusable release is particularly appealing.

We ablate the choice of representation in Section G.4, comparing the Chebyshev moments and Gram block used by NoisyMomentPFN and MomentPFN with either block alone and with monomial moments, where we pretrain a separate model on the same prior for each representation. As shown in Figure 12, the combined Chebyshev and Gram representation performs best on every stratum and on both RMSE and CRPS. In particular, monomial moments perform substantially worse: the two bases span the same degree-k polynomials, but the change of basis from monomial to Chebyshev moments is ill-conditioned, so isotropic noise on monomial moments erases the high-order components that the Chebyshev release retains at noise $O(\sqrt{j})$ (Musco et al., 2025).

Section G.5 removes privacy noise altogether and studies the noise-free limit of the representation. In Figure 13, MomentPFN is already competitive with non-private ridge, and we find that the pairwise moment block accounts for the improvement beyond what is available from the Gram block alone. Section G.6 and Figure 14 then study the effect of noise curriculum in pretraining, at inference on noisy and on clean releases.

5 Conclusion

We introduced NoisyMomentPFN, a tabular foundation model that predicts from a DP release rather than from the training rows themselves. The release combines sliced Chebyshev moments of pairs of feature projections and the target with a Gram block, achieving privacy with a single call to the Gaussian mechanism. A model pretrained directly on these noisy releases can then answer any number of queries by post-processing, without retraining and incurring no additional privacy budget.

Across our regression benchmarks, NoisyMomentPFN improves on DP ridge at nearly every privacy budget in aggregate and produces better predictive distributions than a per-table optimized MLP at moderate and large budgets. It is also more accurate than feeding a private synthetic table to a non-private foundation model, while making predictions from one release in milliseconds. Future work could explore alternative moment statistic choices, attempt to scale our model, and address the remaining point prediction gap, where the DP-SGD MLP is stronger, particularly on small tables.

AI USE STATEMENT

In this work, we used generative AI tools to help implement our training pipeline, specifically the PFN pretraining and evaluation code, which builds on the open-source TabPFN v1 implementation that we credit. We have not used generative AI tools for idea formation, research methodology or experimental design, or formalizing / theory-related tasks. We have reviewed all AI-assisted work, e.g., the paper was written by the authors, the code was checked by the authors, and all plots were produced by the authors directly from experimental results, which we verified. We take responsibility for the final content of this work, including artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops a method for making predictions from sensitive tabular data under differential privacy, a privacy-protective goal. Every table used in the experiments is public and was released by its original curators for research use.

REPRODUCIBILITY STATEMENT

The supplementary code release contains the DP mechanism, the prior, the pretraining and prediction code of MomentPFN and NoisyMomentPFN, the two baselines we implemented, and a benchmark harness. Additionally, our paper contains exhaustive details to help with reproducing the performance described within.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016. doi: 10.1145/2976749.2978318.

Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. 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. 394–403. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/balle18a.html.

Matej Balog, Ilya Tolstikhin, and Bernhard Schölkopf. Differentially private database release via kernel mean embeddings. In *International Conference on Machine Learning (ICML)*, pp. 414–422, 2018. URL https://proceedings.mlr.press/v80/balog18a.html.

March Boedihardjo, Thomas Strohmer, and Roman Vershynin. Private measures, random walks, and synthetic data. *Probability Theory and Related Fields*, 189(1–2):569–611, 2024. doi: 10.1007/s00440-024-01279-z.

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

Vladimir Braverman, Aditya Krishnan, and Christopher Musco. Sublinear time spectral density estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 1144–1157, 2022. doi: 10.1145/3519935.3520009.

Francesco Capano and Jonas Böhler. Probing memorization of tabular in-context learning. In *ICML 2026 Workshop on Foundation Models for Structured Data*, 2026. URL https://arxiv.org/abs/2606.31208. arXiv preprint arXiv:2606.31208.

Alycia N. Carey, Karuna Bhaila, Kennedy Edemacu, and Xintao Wu. DP-TabICL: In-context learning with differentially private tabular data. *arXiv preprint arXiv:2403.05681*, 2024.

Kai Chen, Xiaochen Li, Chen Gong, Ryan McKenna, and Tianhao Wang. Benchmarking differentially private tabular data synthesis. *arXiv preprint arXiv:2504.14061*, 2025.

Benjamin Coleman and Anshumali Shrivastava. A one-pass distributed and private sketch for kernel sums with applications to machine learning at scale. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 3252–3265, 2021. URL https://arxiv.org/abs/2006.09352.

Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of stochastic parrots: Differentially private prompt learning for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.15594.

Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. In *Conference on Learning Theory (COLT)*, 2018. URL https://arxiv.org/abs/1803.10266.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2506.16791.

Sebastian Felix Fischer, Liana Harutyunyan, Matthias Feurer, and Bernd Bischl. OpenML-CTR23: A curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023. URL https://openreview.net/forum?id=HebAOoMm94.

James Foulds, Joseph Geumlek, Max Welling, and Kamalika Chaudhuri. On the theory and practice of privacy-preserving Bayesian data analysis. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016. URL https://arxiv.org/abs/1603.07294.

Pieter Gijsbers, Marcos L. P. Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. AMLB: an AutoML benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024. URL https://www.jmlr.org/papers/v25/22-0493.html.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Google Research. Introducing TabFM: A zero-shot foundation model for tabular data. https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/, 2026. Model weights at https://huggingface.co/google/tabfm-1.0.0-pytorch; accessed September 2026.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.02848.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 18932–18943, 2021. URL https://arxiv.org/abs/2106.11959.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.24210.

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*, 2022. URL https://arxiv.org/abs/2207.08815.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Frederik Harder, Kamil Adamczewski, and Mijung Park. DP-MERF: Differentially private mean embeddings with random features for practical privacy-preserving data generation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. URL https://proceedings.mlr.press/v130/harder21a.html.

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. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025. doi: 10.1038/s41586-024-08328-6.

Weihao Kong and Gregory Valiant. Spectrum estimation from samples. *The Annals of Statistics*, 45 (5):2218–2247, 2017.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26 (226):1–16, 2025. URL http://jmlr.org/papers/v26/25-0512.html.

Terrance Liu, Giuseppe Vietri, and Steven Z Wu. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems*, 34: 690–702, 2021.

Terrance Liu, Jingwu Tang, Giuseppe Vietri, and Steven Z. Wu. Generating private synthetic data with genetic algorithms. In *International Conference on Machine Learning (ICML)*, 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)*, 2025. URL https://arxiv.org/abs/2410.18164.

Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pp. 4435–4444. PMLR, 2019.

Ryan McKenna, Gerome Miklau, and Daniel Sheldon. Winning the NIST contest: A scalable and general approach to differentially private synthetic data. *Journal of Privacy and Confidentiality*, 11(3), 2021. arXiv preprint arXiv:2108.04978.

Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. AIM: an adaptive and iterative mechanism for differentially private synthetic data. *Proceedings of the VLDB Endowment*, 15(11): 2599–2612, 2022.

Ilya Mironov. Rényi differential privacy. In *IEEE Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017. URL https://arxiv.org/abs/1702.07476.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Cameron Musco, Christopher Musco, Lucas Rosenblatt, and Apoorv Vikram Singh. Sharper bounds for chebyshev moment matching, with applications. In *Proceedings of the Thirty Eighth Annual Conference on Learning Theory (COLT)*, volume 291 of *Proceedings of Machine Learning Research*, pp. 4309–4358. PMLR, 2025. URL https://proceedings.mlr.press/v291/musco25a.html.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1610.05755.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=rkZB1XbRZ.

Natalia Ponomareva, Zheng Xu, H. Brendan McMahan, Peter Kairouz, Lucas Rosenblatt, Vincent Cohen-Addad, Cristóbal Guzmán, Ryan McKenna, Galen Andrew, Alex Bie, Da Yu, Alex Kurakin, Morteza Zadimoghaddam, Sergei Vassilvitskii, and Andreas Terzis. How to DP-fy your data: A practical guide to generating synthetic data with differential privacy. *Journal of Artificial Intelligence Research*, 86, 2026. arXiv preprint arXiv:2512.03238.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.05564.

Alain Rakotomamonjy and Liva Ralaivola. Differentially private sliced Wasserstein distance. In *International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/rakotomamonjy21a.html.

Ilana Sebag, Muni Sreenivas Pydi, Jean-Yves Franceschi, Alain Rakotomamonjy, Mike Gartrell, Jamal Atif, and Alexandre Allauzen. Differentially private gradient flow based on the sliced wasserstein distance. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=aiOHc1LGpD.

Xinyu Tang, Richard Shin, Huseyin A. Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-preserving in-context learning with differentially private few-shot generation. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.11765.

Toan Tran, Arturs Backurs, Zinan Lin, Victor Reis, Li Xiong, and Sergey Yekhanin. Differentially private synthetic tabular data via private evolution. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2606.08259. arXiv preprint arXiv:2606.08259, titled "Differentially Private Synthetic Data via APIs 4: Tabular Data".

Duy Vu and Aleksandra Slavkovic. Differential privacy for clinical trial data: Preliminary evaluations. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 138–143, 2009. doi: 10.1109/ICDMW.2009.52.

Dariush Wahdany, Matthew Jagielski, Jesse C. Cresswell, Adam Dziedzic, and Franziska Boenisch. TabPATE: Differentially private tabular in-context learning without public data. *arXiv preprint arXiv:2606.31474*, 2026.

Yu-Xiang Wang. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. URL https://arxiv.org/abs/1803.02596.

Tong Wu, Ashwinee Panda, Jiachen T. Wang, and Prateek Mittal. Privacy-preserving in-context learning for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2305.01639.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Trans. Database Syst.*, 42(4), October 2017. ISSN 0362-5915. doi: 10.1145/3134428. URL https://doi.org/10.1145/3134428.

A Related work

Tabular foundation models. Prior-fitted networks train a transformer on datasets sampled from a prior over tasks, so that a single forward pass over in-context training rows approximates the posterior predictive (Müller et al., 2022). TabPFN instantiated this for tabular data with a synthetic prior of structural causal models (Hollmann et al., 2023; 2025), and its successors, TabPFN-2.5 (Grinsztajn et al., 2025), TabICL (Qu et al., 2025), TabDPT (Ma et al., 2025), and TabFM (Google Research, 2026), scale the context, the prior, or the training data. All of them read the raw rows of the table, and recent work shows that rows in context can be tested for membership through the predictions (Wahdany et al., 2026; Capano & Böhler, 2026). In our work, we keep the pretraining paradigm and the transformer backbone but replace the row vocabulary with a fixed-size noisy release.

Private in-context learning and private prediction. For language models, privacy of the examples placed in a prompt has been obtained by noisy aggregation over ensembles of prompts (Duan et al., 2023; Wu et al., 2024) or by generating private synthetic demonstrations (Tang et al., 2024), and DP-TabICL applies the first idea to serialized tabular records (Carey et al., 2024). Closest to our setting, TabPATE (Wahdany et al., 2026) protects tabular in-context learning with a PATE-style ensemble of teachers over disjoint shards of the context (Papernot et al., 2017; 2018), in the tradition of private prediction (Dwork & Feldman, 2018). Our method, however, is a parallel stream of work, as these methods pay a privacy cost per query or per released label, whereas ours pays once per table, after which any number of queries are answered at no further cost, and needs neither an ensemble nor public data from the task at hand.

Differentially private learning on tables. The dominant approach to private learning is DP-SGD, which clips and noises per-example gradients (Abadi et al., 2016), with Renyi or PLD accounting (Mironov, 2017; Gopi et al., 2021) and reference implementations such as Opacus (Yousefpour et al., 2021). Our DP-SGD MLP baseline is trained this way, based loosely on the state-of-the-art performance from the recent TabM model (Gorishniy et al., 2025). For linear models there are far cheaper alternatives based on perturbing sufficient statistics (Vu & Slavkovic, 2009; Foulds et al., 2016), and DP Ridge with the adaptive rule of Wang (2018) is considered as a strong practical baseline.

Another paradigm is to apply public methods to private synthetic data. A large body of work has developed marginal-based generators for DP synthetic data (Zhang et al., 2017; McKenna et al., 2019; 2021; 2022; Liu et al., 2021), as well as genetic and evolutionary approaches (Liu et al., 2023; Tran et al., 2026). See Chen et al. (2025) for a detailed comparison of these methods in terms of downstream performance. In this work, we use AIM as the generator of DP synthetic tables, as it is generally regarded as a state-of-the-art approach, and then perform inference with foundation models by providing the private synthetic rows in context. We use this pipeline as a representative instantiation of the public methods on top of private synthetic data paradigm.

Moment and feature releases. Moment matching recovers a distribution from a few of its generalized moments, and the choice of basis decides how much noise the recovery tolerates. In particular, monomial moments are catastrophically ill-conditioned, whereas Chebyshev moments admit Wasserstein-1 recovery from noisy estimates at a rate that nearly matches the optimal rate for private distribution estimation in one dimension (Kong & Valiant, 2017; Braverman et al., 2022; Musco et al., 2025; Boedihardjo et al., 2024). Moreover, in higher dimensions, matching Chebyshev moments along random projections controls the sliced Wasserstein distance, and the sliced distance itself can be privatized (Rakotomamonjy & Ralaivola, 2021; Sebag et al., 2025). One could also release noisy random-feature or kernel mean embeddings once and fit a generator to them as a one-shot approach (Balog et al., 2018; Harder et al., 2021), as are other related private sketches for downstream learning (Coleman & Shrivastava, 2021). Our use of these statistics is different: we do not reconstruct a distribution from them. Instead, we pretrain a network that reads the noisy moments directly and outputs a predictive distribution, so whatever recovery is needed is learned from the prior.

B Additional methodological details

We provide additional details omitted from Section 3 for brevity.

B.1 Additional details on the setup and the (private) release

We provide more details on the setup and the release, including several design choices. A more comprehensive walkthrough of the private release procedure is deferred to Section C.

Parameters assumed to be public. The parameters $\mathbf{p}$ on which a release depends other than the rows of S are: the sizes n, d, and, for classification, C; the rescaling bounds $(\ell_j, u_j)_{j\le d}$ and (ℓ_y, u_y); the privacy budget (ϵ, δ); the Chebyshev degree k; the number of pairs L and the pairs $(\theta_\ell, \theta'_\ell)$ themselves, drawn from a seeded generator; the projection scale s; and the Gram weight w. We assume $\mathbf{p}$ is fixed without looking at the rows of S, and the guarantee of Proposition 1 is conditional on $\mathbf{p}$. As is standard in the *evaluation* of DP methods, the release treats n and d as public, although real-world deployments may warrant a more stringent treatment of these parameters.

Projections. The choice $s = \sqrt{d}$ never clips, since $|\langle X_i, \theta\rangle| \le \sqrt{d}$ for $X_i \in [-1,1]^d$ and a unit direction, but a smaller s spreads typical projections over more of $[-1,1]$ at the expense of clipping outliers. We use $s = 1$, under which coordinate-axis pairs are never clipped, while about ≈one in ten projections onto random directions are clipped. We emphasize that setting s a priori does not affect the sensitivity bound of Section 3.1, which uses only that $u \in [-1,1]$.

The one-dimensional release. The one-dimensional release is an earlier version of the model that we went away from as attempted to improve regression performance. It is still used by the classification model, by the ablations labeled "1D release", and by an earlier regression model. It replaces pairs of directions with $L = 256$ single directions θ_ℓ (the second direction θ'_ℓ and the middle index are dropped) and the block Equation 1 by the $(k+1)\times(k+1)$ matrix

$$M_\ell[a,b] = \frac{1}{n}\sum_{i=1}^{n} \bar{T}_a(u_{i\ell})\bar{T}_b(Y_i), \tag{7}$$

with the constant entry $M_\ell[0,0] = 1/\pi$ dropped, the weights $s_{ab} = \sqrt{\max(a,1)\max(b,1)}$, and the bounds $\beta_{ab} = \gamma_a\gamma_b$; the sensitivity Equation 3 holds with the sum over $(a,b) \neq (0,0)$, which equals $((1+2H_k)/\pi)^2 - \pi^{-2}$. With $L = 256$ directions and $k = 6$ its sensitivity is approximately equal to that of the pair release with 128 pairs, so the two releases carry about the same noise per entry at the same budget.

B.2 Additional details on the predictor

Target feature map. The regression and classification releases are two instances of one construction in which the target enters the moment block through a feature map $\psi = (\psi_b)_{b\in\mathcal{B}}$: for regression, $\mathcal{B} = \{0,\dots,k\}$ and $\psi_b(y) = \bar{T}_b(y)$; for classification, $\mathcal{B} = [C]$ and $\psi_b(y) = \mathbf{1}[y = b]$. In both cases $\|\psi_b\|_\infty \le 1$, and the target enters the Gram block as $\psi(y) = y$ for regression and as the one-hot vector e_y for classification, so that $z_i = (X_i, 1, \bar{\psi}(Y_i))$. For regression all of this reduces to the definitions of Section 3.

Decoder inputs: closed-form features. We restate the decoder inputs of Section 3.2 and give their closed forms. The ridge feature on the query token is the sufficient statistics perturbation estimator read off the noisy Gram block. Writing $\tilde{G}$ in blocks over $(X,1)$ and Y as $\tilde{G}_{xx} \in \mathbb{R}^{(d+1)\times(d+1)}$, $\tilde{G}_{xy} \in \mathbb{R}^{d+1}$, and $\tilde{G}_{yy}$, we have

$$\hat{\beta} = (\tilde{G}^{+}_{xx} + \lambda I)^{-1}\tilde{G}_{xy}, \qquad \hat{y}(x) = \hat{\beta}^\top(x,1), \qquad \hat{\tau}^2 = \max\left(10^{-4},\ \tilde{G}_{yy} - \hat{\beta}^\top\tilde{G}_{xy}\right), \tag{8}$$

where $\tilde{G}^{+}_{xx}$ is the noisy, possibly indefinite $\tilde{G}_{xx}$ with its eigenvalues clamped to be at least 10^{-6} and $\lambda = \sqrt{d+1}\sigma/w$; $\hat{\tau}^2$ is the released residual second moment of the fit (the least-squares form, used as is for the ridge $\hat{\beta}$), and the query token carries $\hat{y}(x)$ clipped to $[-1,1]$ and $\log\hat{\tau}$.

Each pair token is summed with the embedding of the feature vector computed from its block $\tilde{M}_\ell$ and the query's two projections $u_\ell(x)$ and $u'_\ell(x)$: the bases, the marginal estimate, and the truncated series $\hat{f}_\ell(x,y)$ of Equation (5) on a fixed grid of $y \in [-1,1]$. By the orthonormality of the basis, $\hat{f}_\ell$ is the degree-k estimate of the joint density of (u_ℓ, u'_ℓ, Y) divided by the product Chebyshev weight.

Up to the factor $(1-y^2)^{-1/2}$ and normalization over y, it is the raw moment-matching estimate of the conditional distribution of the target given the two projections; it may be negative, which the network learns to handle. Likewise, the marginal estimate $\sum_{a,b} \tilde{M}_\ell[a,b,0]\bar{T}_a(u_\ell(x))\bar{T}_b(u'_\ell(x))$ is the same estimate one order lower, the degree-k estimate of the joint density of the two projections at the query divided by the product Chebyshev weight.

B.3 Additional details on the pretraining

Formally, the pretraining samples are drawn as follows, for $i = 1, 2, \ldots$,

$$P_i \sim \Pi, \qquad \mathbf{p}_i \sim \mathcal{P}, \qquad S_i \sim P_i^{n_i}, \qquad (X_{ij}, Y_{ij}) \overset{\text{i.i.d.}}{\sim} P_i, j \in [K], \qquad \tilde{R}_i = \mathcal{M}_{\mathbf{p}_i}(S_i; \xi_i), \tag{9}$$

where the K query rows are independent of S_i given P_i and $\mathcal{P}$ is a distribution over public parameters. The dimension d_i is a property of P_i (the prior draws it first), the rescaling bounds of the empirical-range regime are computed from S_i, and the argument below uses only that Y_{ij} is independent of $(\tilde{R}_i, \mathbf{p}_i)$ given (P_i, X_{ij}).

The minimizer of the pretraining objective For each fixed $(x, \tilde{R}, \mathbf{p})$ the objective Equation 6 is the expected cross-entropy of $q_\phi(\cdot \mid x, \tilde{R}, \mathbf{p})$ against the conditional law of Y given $(X = x, \tilde{R}, \mathbf{p})$ under Equation 9, so its minimizer over all conditional distributions is

$$q^\star(y \mid x, \tilde{R}, \mathbf{p}) = \Pr\big(Y = y \mid X = x, \tilde{R}, \mathbf{p}\big) = \mathbb{E}\big[P(Y = y \mid X = x)\big| X = x, \tilde{R}, \mathbf{p}\big], \tag{10}$$

where the outer expectation is over the posterior of P given the release, the public parameters, and the query feature (for regression, the same statement holds for the binned target). The conditioning on $X = x$ is not vacuous, since the query feature is itself drawn from P and carries information about P.

The prior over tasks. We provide more details on the prior Π over tasks, a mixture of synthetic and real priors, and the pretraining data generation process below. Three quarters of tables come from a synthetic prior, and within it, half of the tables come from a structural causal prior in the style of TabPFN: a random directed acyclic graph whose root nodes draw from assorted distributions and whose inner nodes are random functions of their parents (small networks with sharp activations, decision trees, products, steps, and periodic maps) with a little noise, the target being an inner node and the features other nodes, upstream or downstream of it. The other half of the synthetic tables have features generated by the random-network and Gaussian-process generators of the TabPFN prior (Hollmann et al., 2023) and a target that is, with probabilities 0.25, 0.15, 0.18, and 0.42, a random ensemble of shallow trees, a linear map with sparse pairwise interactions and a smooth nonlinearity, a sparse linear map, or the generator's own output. To reproduce public tables, the prior then applies augmentations producing few-valued, binary, skewed, censored, and nominal columns, missing cells, and skewed or discrete targets. The remaining quarter of tables are subsampled in rows and columns from 36 public regression tables disjoint from every evaluation table.

The prior over public parameters. The prior $\mathcal{P}$ of public parameters is generated in the following ways: draws $n_i \sim \mathrm{LogUniform}[10^2, 10^4]$, $d_i \sim \mathrm{Uniform}\{2, \ldots, 20\}$, $\epsilon_i \sim \mathrm{LogUniform}[0.1, 10]$, and $\delta_i = n_i^{-2}$, with fresh pairs per table. The rescaling bounds of a synthetic table are drawn from three regimes, the empirical range of the table, clipped percentiles of it, or a range wider than the data, so that the model does not learn to assume that the data fills its box. The remaining constants are fixed: $L = 128$, $k = 6$, $w = 5$, $s = 1$, $B = 100$, $K = 64$.

Optimization. The clean statistics $R(S_i)$ of each table are computed once and cached; a cache of $80,000$ tables is regenerated every $2,500$ steps. At every optimization step, for each table in the batch, the budget ϵ_i is redrawn, a uniformly random subset of between $L/2$ and L of its pairs is kept, the sensitivity Δ is recomputed for the number of pairs kept, and fresh noise ξ_i is added. One cached table thus yields many distinct releases, and no noisy release is seen twice. The encoder has 8 layers and the decoder 4, with width 384, 6 heads, and feed-forward width 1536, for 21.7M parameters. Tables with fewer than $d_{\max} = 20$ columns are zero-padded to $d_{\max}$ in the directions, the Gram triangle, and the query, and $\mathbf{p}$ enters every token as $(\log \epsilon, \log \delta, \log n, d)$; the decoder attends to the

Gram token as well as to the pair tokens. The reported regression model was initialized from a model trained for 60k steps on clean releases and trained for 60k more steps on the noisy curriculum, on batches of 64 tables with 64 held-out queries each, with AdamW at peak learning rate 10^{-4}, $1,000$ warmup steps, cosine decay, weight decay 0.01, and an exponential moving average of the weights with decay 0.9995. Pretraining takes about four GPU hours per stage on one accelerator.

Necessary modifications for classification. The classification model uses the one-dimensional release with C classes, $\psi_b(y) = \mathbf{1}[y = b]$, so M_ℓ is the $(k+1) \times C$ matrix whose column b holds the Chebyshev moments of the projections of the rows in class b, summed over those rows and divided by n rather than by the class count. Its row $a = 0$ is $1/\sqrt{\pi}$ times the class proportions and is released, so no entry is constant and none is dropped. In the Gram block, $\bar{\psi}(Y_i) = e_{Y_i}$, so $z_i \in \mathbb{R}^{d+1+C}$ and $N_G = (d+1+C)(d+2+C)/2$. The classes carry no order, so they are weighted equally, $s_{ab} = \sqrt{\max(a,1)}$. Replacing one row either moves two class columns by at most γ_a/n each, if the classes differ, or one column by at most $2\gamma_a/n$, if they agree (and the $a = 0$ entry not at all), so the moment part of the squared sensitivity is $L\big(2\gamma_0^2 + 4\sum_{a\geq 1}\gamma_a^2/a\big)/n^2$. The predictor is a categorical distribution over $[C]$ with C class logits, whose argmax is the class prediction. iI the closed-form features, $\tilde{G}_{xy} \in \mathbb{R}^{(d+1)\times C}$, so that $\hat{y}(x)$ is a vector of C scores, and per direction the per-class series $\sum_a \tilde{M}_\ell[a,b]\bar{T}_a(u_\ell(x))$ for $b \in [C]$ replaces the grid of Equation 5. In pretraining, classification tables are formed by cutting a regression target into intervals or by clustering the rows around random prototypes, with two classes half of the time, and the head covers at most ten classes.

C MATERIALIZING THE FULL RELEASE

Algorithm 1 states the release end to end. Since everything else in the paper is a function of the vector to which the Gaussian mechanism is applied, we write that vector out explicitly here.

Algorithm 1 RELEASE$(S, \mathbf{p})$

1: $u_{i\ell} \leftarrow \mathrm{clip}(\langle X_i, \theta_\ell\rangle/s, -1, 1)$ and $u'_{i\ell} \leftarrow \mathrm{clip}(\langle X_i, \theta'_\ell\rangle/s, -1, 1)$ for all $i \in [n]$, $\ell \in [L]$
2: compute $M_1, \ldots, M_L$ and G by Equation 1; form $v(S)$ by Equation 2
3: $\sigma \leftarrow \sigma_{\mathrm{AG}}(\Delta, \epsilon, \delta)$ with Δ from Equation 3
4: $\tilde{v} \leftarrow v(S) + \sigma\xi, \quad \xi \sim \mathcal{N}(0, I)$
5: **return** $\tilde{R} = (\tilde{M}_1, \ldots, \tilde{M}_L, \tilde{G})$, the re-weighted entries of $\tilde{v}$ with the constant entry filled in

Fix a pair ℓ. The rows of S projected onto the pair, paired with their targets, form a sample $\{(u_{i\ell}, u'_{i\ell}, Y_i)\}_{i\leq n}$ from a distribution on $[-1,1]^3$, and its degree-k Chebyshev moments in the product basis $\{\bar{T}_a \otimes \bar{T}_b \otimes \bar{T}_c\}$ are the entries of the tensor $M_\ell \in \mathbb{R}^{(k+1)^3}$ of Equation 1. The slice $c = 0$ holds the joint moments of the two projections alone, which describe where the rows lie along the pair; the fibers $(0, 0, c)$ hold the moments of the target alone, which do not depend on ℓ; and the interior holds the cross moments that carry the dependence of Y on (u_ℓ, u'_ℓ). The $\mathbf{0}$ entry is $\pi^{-3/2}$ for every table and is not released. The fibers $(0, 0, c)$ do not depend on ℓ, so the release holds L noisy copies of the target moments, and axis pairs that share a direction duplicate its marginal moments; these entries cost about two percent of the moment sensitivity, and we keep them so that every block has the same layout. Dividing entry (a, b, c) by s_{abc} and stacking the entries of all L pairs, in lexicographic order within each pair, gives the moment part of the released vector, $v_M(S) \in \mathbb{R}^{L[(k+1)^3-1]}$; on the fibers $(a, 0, 0)$ the weights reduce to the $1/\sqrt{a}$ weights of the one-dimensional mechanism of Musco et al. (2025).

The moments describe the target along one pair at a time. To give the model a global linear view of the table, the release also contains the second-moment matrix of the augmented row $z_i = (X_i, 1, Y_i)$,

$$G = \frac{1}{n}\sum_{i=1}^{n} z_i z_i^\top = \begin{bmatrix} \frac{1}{n}\sum_i X_i X_i^\top & \frac{1}{n}\sum_i X_i & \frac{1}{n}\sum_i X_i Y_i \\ \frac{1}{n}\sum_i X_i^\top & 1 & \frac{1}{n}\sum_i Y_i \\ \frac{1}{n}\sum_i Y_i X_i^\top & \frac{1}{n}\sum_i Y_i & \frac{1}{n}\sum_i Y_i^2 \end{bmatrix}, \tag{11}$$

the sufficient statistic of ridge regression of Y on $(X, 1)$: the top-left block is the uncentered feature covariance, the middle row and column the feature means, the last row and column the feature-target

cross moments, and the bottom-right entry the second moment of the target. Since G is symmetric, its upper triangle, scaled by the block weight w, forms $v_G(S) \in \mathbb{R}^{N_G}$ with $N_G = (d+2)(d+3)/2$; the constant entry $G_{d+1,d+1} = 1$ has zero sensitivity and is counted in N_G only to keep the bound simple. The two blocks are concatenated into a single vector and noised once,

$$v(S) = \begin{bmatrix} v_M(S) \\ v_G(S) \end{bmatrix} \in \mathbb{R}^{L[(k+1)^3-1]+N_G}, \qquad \tilde{v} = v(S) + \sigma\xi, \quad \xi \sim \mathcal{N}(0, I), \tag{12}$$

with $\sigma = \sigma_{\mathrm{AG}}(\Delta, \epsilon, \delta)$ calibrated to the sensitivity Equation 3 of the whole vector. Since every entry of $v(S)$ is an average over rows of a function bounded by β_{abc}/s_{abc} or by w, the sensitivity has the closed form Equation 3 in (n, d, k, L, w), and no composition across blocks or pairs is needed. Undoing the weights and restoring the constant entry gives the release the model reads, Equation 4. With $L = 128$, $k = 6$, and $d = 20$, the release has $128 \cdot 342 + 253 = 44,029$ numbers, whatever the value of n.

D Evaluation protocol

In the idealized setting, the evaluation task is a fresh draw from the pretraining hierarchy,

$$P_{\mathrm{eval}} \sim \Pi, \qquad S_{\mathrm{train}} \sim P_{\mathrm{eval}}^{n}, \qquad (X_*, Y_*) \sim P_{\mathrm{eval}}, \tag{13}$$

with (X_*, Y_*) independent of S_{train} given P_{eval}, and a method is scored by a proper scoring rule of its predictive distribution at X_* against Y_*. Under this assumption the pretraining objective Equation 6 is the expected evaluation loss under the log score, and the Bayes predictor Equation 10 that the trained network approximates minimizes the expected score under every proper scoring rule, the CRPS included. In our experiments P_{eval} is instead the empirical distribution of a real table fixed in advance and disjoint from every table used in pretraining, i.e., the assumption $P_{\mathrm{eval}} \sim \Pi$ does not hold, and thus the results measure how far the learned prior transfers.

Privacy budgets are $\epsilon \in \{0.5, 1, 2, 4, 8\}$ with $\delta = n^{-2}$, each with three independent releases (two on the large suite at 50k rows, five on classification), and the OpenML-CTR23 tables with five random splits. Following the convention of the benchmark and of the baselines we compare to, the rescaling bounds are the minimum and maximum of each column over all rows of the table, training and test, which treats the empirical range as public schema information; the mechanism itself is agnostic to how the bounds are chosen, and the guarantee of Proposition 1 is exact whenever they are public. Paired comparisons between methods count the tables on which one method has the lower mean score.

E Tables

The numeric and mixed suites are drawn from the fixed-split regression tables of TALENT (Liu et al., 2025) with at most 20 columns and at most 10,000 rows; seven further tables are drawn from OpenML-CTR23 (Fischer et al., 2023); the large suite is drawn from OpenML suites 336 and 335 of Grinsztajn et al. (2022), suite 457 of Erickson et al. (2025), and suite 269 of Gijsbers et al. (2024), and Tables 1 and 2 list every table with its number of rows and column types. The classification suite holds 33 tables from TALENT with 2 to 10 classes. For reporting, the OpenML-CTR23 tables are merged into the numeric and mixed strata according to whether they contain categorical columns, so that each stratum groups tables of one column type: numeric (17), mixed (22), and large (13).

F The DP-SGD baseline

Architecture and training. The network has three hidden layers of 512 units with ReLU and dropout 0.1. We train it with AdamW at learning rate 10^{-3} for 50 epochs, using Poisson batches of expected size $\min(256, n/8)$, per-example gradient clipping at norm one, and Gaussian noise on the summed gradient via the Opacus 0.14 privacy engine (Yousefpour et al., 2021). The Gaussian-head variant outputs a mean and a log-variance and is trained with the Gaussian negative log-likelihood under the same mechanism. The non-private MLP used throughout (e.g., of Figure 3) has the same architecture and hyperparameters, without clipping or noise.

Table 1: The 52 regression tables: source (TALENT, or OpenML with the dataset id), stratum, number of rows, and number of numeric (num.) and categorical (cat.) columns. The TALENT tables are scored on their fixed split, the OpenML-CTR23 tables on five random splits, and the large tables on one test split of $2,000$ rows with the training set truncated to 50k rows (Section D).

table	source	stratum	rows	num.	cat.
qsar_aquatic_toxicity	TALENT	numeric	546	8	0
stock	TALENT	numeric	950	9	0
concrete_compressive_strength	TALENT	numeric	1,030	8	0
1000-Cameras-Dataset	TALENT	numeric	1,038	10	0
weather_izmir	TALENT	numeric	1,461	9	0
airfoil_self_noise	TALENT	numeric	1,503	5	0
Wine_Quality_red	TALENT	numeric	1,599	11	0
debutanizer	TALENT	numeric	2,394	7	0
space_ga	TALENT	numeric	3,107	6	0
analcatdata_supreme	TALENT	numeric	4,052	7	0
Laptop_Prices_Dataset	TALENT	numeric	4,441	8	0
Wine_Quality_white	TALENT	numeric	4,898	11	0
cpu_small	TALENT	numeric	8,192	12	0
combined_cycle_power_plant	TALENT	numeric	9,568	4	0
forest_fires	CTR23 44962	numeric	517	10	0
cars	CTR23 44994	numeric	804	17	0
sarcos	CTR23 44976	numeric	48,933	20	0
housing_price_prediction	TALENT	mixed	545	5	7
qsar_fish_toxicity	TALENT	mixed	908	4	2
archive2	TALENT	mixed	1,143	11	1
socmob	TALENT	mixed	1,156	1	4
garments_worker_productivity	TALENT	mixed	1,197	8	5
healthcare_insurance_expenses	TALENT	mixed	1,338	3	3
Is-this-a-good-customer	TALENT	mixed	1,723	9	4
Shop_Customer_Data	TALENT	mixed	2,000	4	2
mauna-loa-atmospheric-co2	TALENT	mixed	2,225	5	1
NHANES_age_prediction	TALENT	mixed	2,277	4	3
Mobile_Phone_Market_in_Ghana	TALENT	mixed	3,600	4	10
Data_Science_Salaries	TALENT	mixed	3,755	1	4
Abalone_reg	TALENT	mixed	4,177	7	1
stock_fardamento02	TALENT	mixed	6,277	5	1
shrutime	TALENT	mixed	10,000	4	6
Kaggle_bike_sharing_demand_challange	TALENT	mixed	10,886	3	6
Shipping	TALENT	mixed	10,999	5	4
Intersectional-Bias-Assessment	TALENT	mixed	11,000	14	5
solar_flare	CTR23 44966	mixed	1,066	2	8
Moneyball	CTR23 41021	mixed	1,232	8	6
health_insurance	CTR23 44993	mixed	22,272	4	7
video_transcoding	CTR23 44974	mixed	68,784	16	2
sulfur	OpenML 44145	large	10,081	6	0
Brazilian_houses	OpenML 44062	large	10,692	8	3
MiamiHousing2016	OpenML 44147	large	13,932	13	0
houses	OpenML 46934	large	20,640	8	0
house_sales	OpenML 44066	large	21,613	15	2
house_16H	OpenML 574	large	22,784	16	0
Food_Delivery_Time	OpenML 46928	large	45,451	6	3
physiochemical_protein	OpenML 46949	large	45,730	9	0
seattlecrime6	OpenML 45043	large	52,031	2	2
diamonds	OpenML 44059	large	53,940	6	3
medical_charges	OpenML 44146	large	163,065	3	0
particulate-matter-ukair-2017	OpenML 44068	large	394,299	3	3
nyc-taxi-green-dec-2016	OpenML 44065	large	581,835	9	7

Accounting. We calibrate the noise multiplier with ϵ at $\delta = 1/n^2$ using RDO accountant (Mironov, 2017), computed with the `dp_accounting` library. We also note that the accountant uses add-or-remove-one adjacency, which is weaker than the replace-one adjacency of our release and of DP ridge at the same ϵ, which favors the DP-SGD MLP.

Hyperparameters. Width, depth, learning rate, and the number of epochs were chosen on the numeric suite at $\epsilon = 1$ and 8 from a grid over widths 64 to 1024, one to four layers, learning rates $3 \cdot 10^{-4}$ to $3 \cdot 10^{-3}$, and 20 to 150 epochs, then frozen for all other experiments. We also tried the parameter-efficient ensembling of TabM (Gorishniy et al., 2025) with 8 to 32 members, which was worse.

Table 2: The 33 classification tables: source, number of rows, numeric (num.) and categorical (cat.) columns, and classes.

table	source	rows	num.	cat.	classes
Pima_Indians_Diabetes_Database	TALENT	768	8	0	2
vehicle	TALENT	846	18	0	4
maternal_health_risk	TALENT	1,014	6	0	3
Basketball_c	TALENT	1,340	11	0	2
banknote_authentication	TALENT	1,372	4	0	2
yeast	TALENT	1,484	8	0	10
wine-quality-red	TALENT	1,599	4	0	6
FOREX_audjpy-day-High	TALENT	1,832	10	0	2
drug_consumption	TALENT	1,884	12	0	7
mfeat-morphological	TALENT	2,000	6	0	10
National_Health_and_Nutrition_Health_Survey	TALENT	2,278	7	0	2
segment	TALENT	2,310	17	0	7
Pumpkin_Seeds	TALENT	2,500	12	0	2
wine	TALENT	2,554	4	0	2
Water_Quality_and_Potability	TALENT	3,276	8	0	2
rice_cammeo_and_osmancik	TALENT	3,810	7	0	2
Gender_Gap_in_Spanish_WP	TALENT	4,746	13	0	3
Wilt	TALENT	4,821	5	0	2
wine-quality-white	TALENT	4,898	11	0	7
sylvine	TALENT	5,124	20	0	2
kdd_ipums_la_97-small	TALENT	5,188	20	0	2
phoneme	TALENT	5,404	5	0	2
page-blocks	TALENT	5,473	10	0	5
delta_ailerons	TALENT	7,129	5	0	2
ringnorm	TALENT	7,400	20	0	2
twonorm	TALENT	7,400	20	0	2
eye_movements_bin	TALENT	7,608	20	0	2
water_quality	TALENT	7,996	20	0	2
JapaneseVowels	TALENT	9,961	14	0	9
BLE_RSSI_dataset_for_Indoor_localization	TALENT	9,984	3	0	3
artificial-characters	TALENT	10,218	7	0	10
pendigits	TALENT	10,992	16	0	10
mammography	TALENT	11,183	6	0	2

G Additional results

G.1 Results by stratum

Figure 5 repeats the layout of Figure 1 over all 52 regression tables, the large suite at its 50k row size, for the three private methods; Figure 6 adds the coverage of the central 90% interval as a third panel, and Figures 7 to 9 split the aggregate into the numeric, mixed, and large strata. Pooled over the 52 tables, NoisyMomentPFN improves over DP ridge at every budget (0.927 against 1.075 at $\epsilon = 0.5$, 0.724 against 0.747 at $\epsilon = 8$, winning 29 to 39 tables), the DP-SGD MLP keeps the lowest RMSE (0.806 to 0.689), and on CRPS NoisyMomentPFN is level with the DP-SGD MLP at $\epsilon = 1$ and ahead from $\epsilon = 2$ (0.342 against 0.379 at $\epsilon = 8$, 30 of 52 tables); its coverage stays at 0.75 to 0.77 across budgets while the DP-SGD MLP's falls from 0.81 to 0.66 and DP ridge's rises from 0.48 to 0.84.

RMSE. NoisyMomentPFN is ahead of DP ridge at every budget on the numeric and mixed strata (by 0.11 and 0.10 at $\epsilon = 1$), and on the large stratum up to $\epsilon = 2$ (Section G.3). The DP-SGD MLP is ahead of NoisyMomentPFN by 0.13, 0.08, and 0.05 on the three strata at $\epsilon = 1$, and by 0.08 and 0.05 on the numeric and large tables at $\epsilon = 8$. On the mixed tables, NoisyMomentPFN is ahead at $\epsilon = 8$ (0.788 against 0.797).

CRPS. NoisyMomentPFN beats DP ridge on 14 to 17 of the 17 numeric tables, 18 to 21 of the 22 mixed tables, and 8 to 11 of the 13 large tables. Against the Gaussian-head DP-SGD MLP it is level on the numeric tables, ahead on the mixed tables from $\epsilon = 1$ on (0.358 against 0.437 at $\epsilon = 8$), and ahead on the large tables at every budget.

Coverage. This is where the methods differ most. NoisyMomentPFN covers 0.83 to 0.84 of the numeric test targets, 0.63 to 0.66 of the mixed ones, and 0.86 to 0.87 of the large ones, at every budget. The Gaussian head DP-SGD MLP falls from about 0.8 to 0.6 or 0.7 as the budget grows, on every stratum, and DP ridge rises from 0.28 to 0.75 on the numeric tables. In other words, NoisyMomentPFN is over-confident by a constant amount that does not depend on the budget, while the

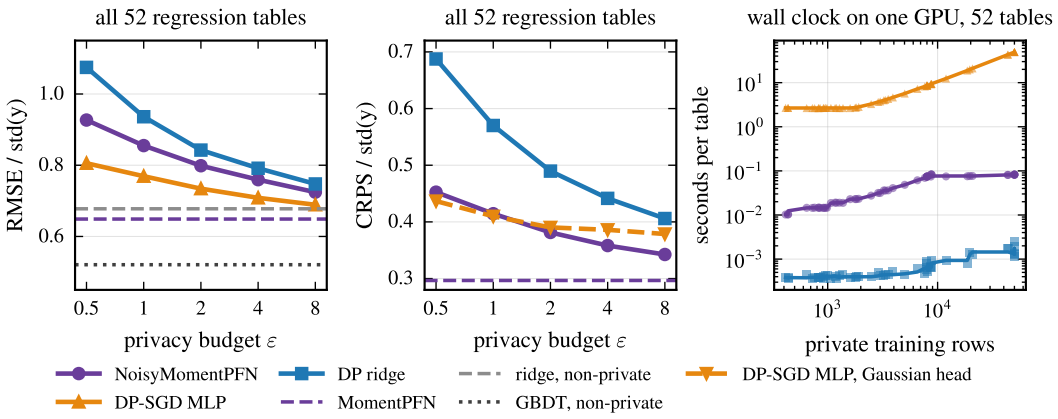


Figure 5: All 52 regression tables: normalized RMSE (left) and CRPS (middle) against the privacy budget, with the non-private references and the noise-free MomentPFN, and the measured wall clock per table on one GPU (right), one point per table with a running median.

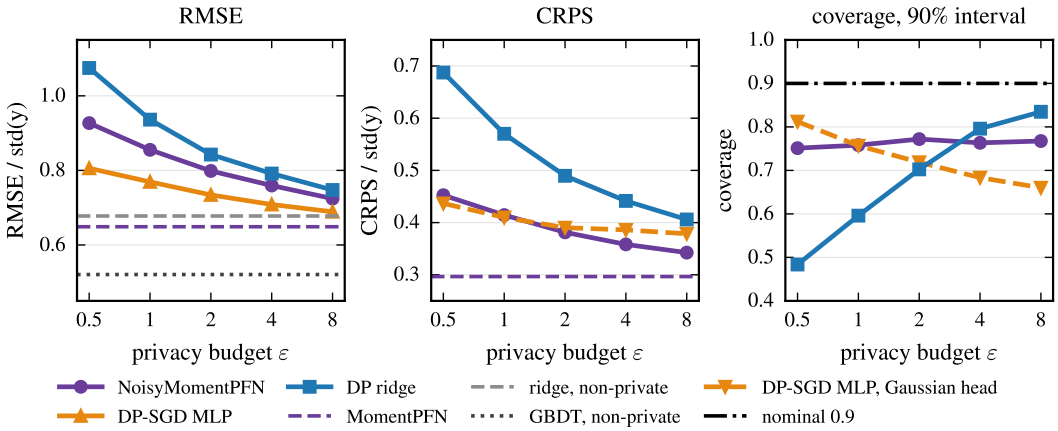


Figure 6: Utility on the 52 regression tables against the privacy budget, averaged over tables. Left: normalized RMSE; middle: normalized CRPS (lower is better); right: empirical coverage of the central 90% interval (nominal 0.9, dash-dotted). Dashed and dotted lines are non-private references fitted on the raw rows, and the dashed purple line is MomentPFN trained and scored without privacy noise, the floor of the representation.

miscalibration of both baselines is budget-dependent. On the numeric and large strata the shortfall is 0.03 to 0.07; on the mixed stratum it is 0.24 to 0.27, and there NoisyMomentPFN is clearly miscalibrated. We attribute this to categorical columns, which random projections of integer codes represent poorly. A predictor that is Bayes-optimal under Π is calibrated on tasks from Π, so the shortfall is likely due to this prior and not the noise for privacy.

G.2 Classification

Figure 10 reports the mean accuracy on the 33 classification tables, five noise draws per table and budget. NoisyMomentPFN is ahead of the DP one-vs-rest ridge at every budget, by 0.03 at $\epsilon = 0.5$ and by less than 0.01 at $\epsilon = 8$, and behind the DP-SGD MLP by 0.01 to 0.03; both private networks remain 0.03 to 0.18 below non-private logistic regression and boosting.

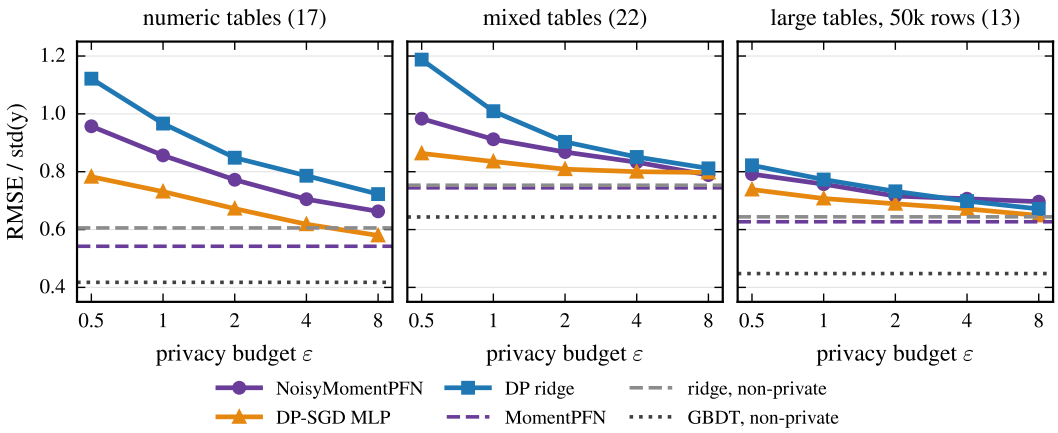


Figure 7: Normalized RMSE against the privacy budget on the numeric (left, 17 tables), mixed (middle, 22 tables), and large (right, 13 tables at 50k training rows) strata; lines and references as in Figure 6.

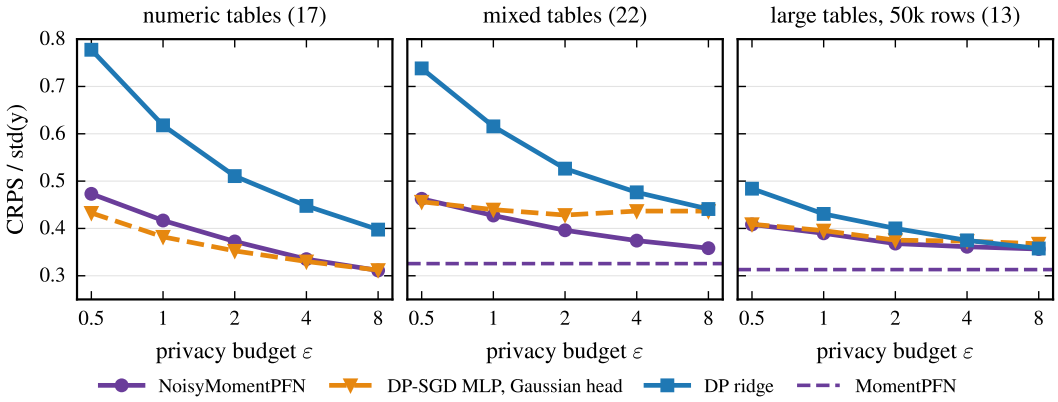


Figure 8: Normalized CRPS against the privacy budget on the three strata, same protocol as Figure 7; the DP-SGD MLP line is its Gaussian-head variant.

G.3 The large suite at 10k and 50k rows

Figure 11 scores the 13 large tables with the training set truncated to 10k and to 50k rows. We expect large tables to favor a reusable release, since the noise in the release shrinks with n while the cost of DP-SGD grows with it. With 50k rows, NoisyMomentPFN reaches 0.757 at $\epsilon = 1$ and 0.696 at $\epsilon = 8$, within 0.05 of the DP-SGD MLP at both budgets. On CRPS it is the best of the three at every budget (0.357 against 0.368 for the Gaussian head at $\epsilon = 8$), with coverage of 0.86 to 0.87. DP ridge overtakes NoisyMomentPFN on these tables at $\epsilon = 4$ and 8 (0.671 against 0.696 at $\epsilon = 8$). This is not surprising: many of these tables are close to linear, and at 50k rows the noise on the Gram matrix is small. Note also that the prior only contains tables of up to 10k rows, so the 50k-row results test how well the size token extrapolates. Section G.6 discusses why the private model does not gain more from the fivefold reduction in noise.

G.4 Ablation of the release

We compare the representation used by NoisyMomentPFN, Chebyshev moments together with the Gram block, against four variants: the Gram block alone, the Chebyshev moments alone, monomial moments alone, and monomial moments with the Gram block. Each variant is pretrained from

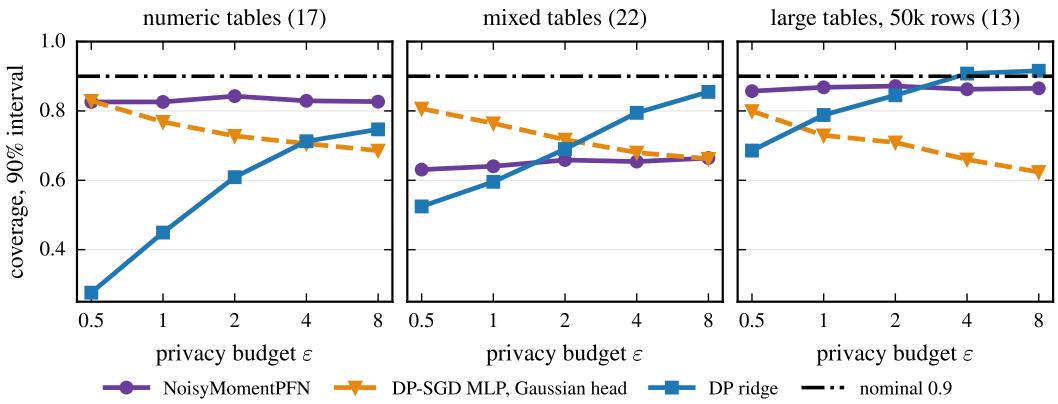


Figure 9: Empirical coverage of the central 90% interval against the privacy budget on the three strata (nominal 0.9, dash-dotted).

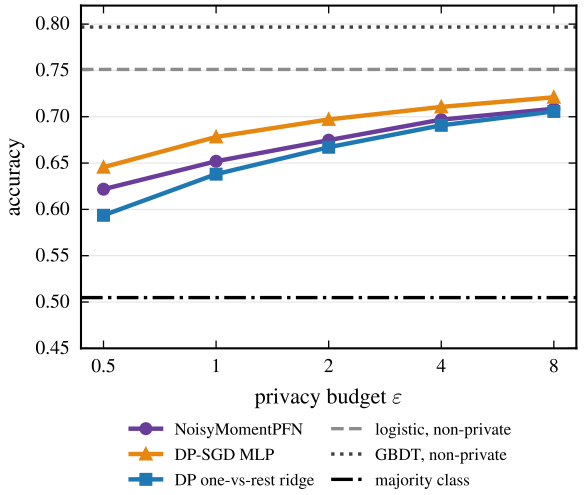


Figure 10: Mean accuracy on the 33 classification tables against the privacy budget, with non-private logistic regression and boosting and the majority class as references.

scratch on the same prior, architecture, noisy curriculum, and 60k-step schedule in its own representation, with the sensitivity of the ablated release, and at inference receives only the corresponding statistics. Figure 12 and Tables 3 and 4 show that the full release is best on every stratum and both metrics. The Gram block alone is close on the mixed and large tables, where the tables are near linear, and the Chebyshev moments alone are close on the numeric tables at small budgets, so the two blocks carry different information. Monomial moments are far worse, alone or with the Gram block.

G.5 The floor of the representation

To separate the cost of the noise from the cost of the representation, we trained the same network on clean releases and scored it on clean releases (this is of course not private). Figure 13 shows the result on the numeric and mixed strata. With one-dimensional slices, the noise-free model lands at about the level of non-private ridge regression: 0.573 against 0.606 on the numeric tables and 0.762 against 0.753 on the mixed ones, which is 0.12 to 0.16 above non-private gradient boosting. We tried several ways of moving this floor, including doubling the number of directions, raising the polynomial degree to 12 or 20, a network with 2.7 times the parameters, the structural causal prior

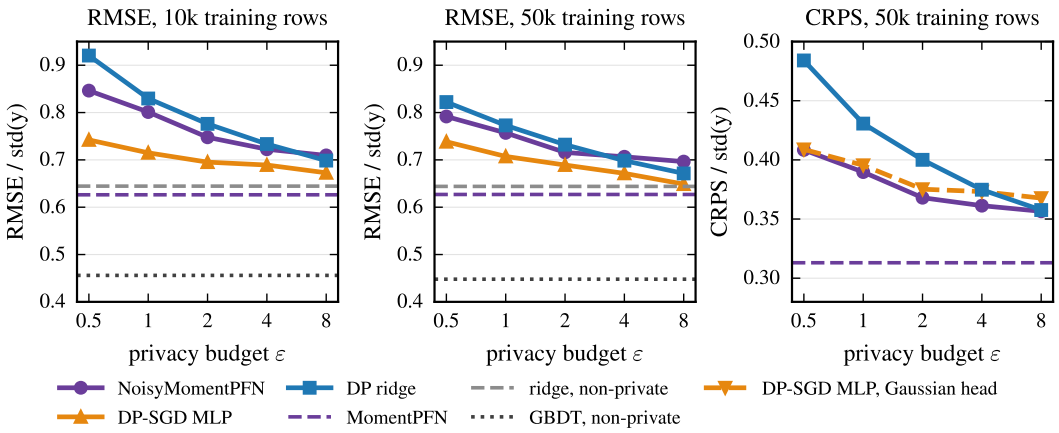


Figure 11: The large suite (13 tables): normalized RMSE with 10k (left) and 50k (middle) training rows, and normalized CRPS with 50k rows (right), one fixed test split of $2,000$ rows per table.

Table 3: Release ablations of NoisyMomentPFN: mean normalized RMSE at $\varepsilon = 1$ and $\varepsilon = 8$ on each suite (tables per suite: numeric 17, mixed 22, large, 50k rows 13). Every model shares the architecture, prior, noisy curriculum and 60k step schedule; only the released statistics differ. Best per column in bold.

	numeric		mixed		large, 50k rows	
release	$\varepsilon = 1$	$\varepsilon = 8$	$\varepsilon = 1$	$\varepsilon = 8$	$\varepsilon = 1$	$\varepsilon = 8$
Gram + Chebyshev moments (NoisyMomentPFN)	**0.825**	**0.654**	**0.927**	**0.792**	**0.733**	**0.663**
Chebyshev moments only	0.866	0.684	0.941	0.881	0.790	0.793
Gram block only	0.875	0.666	0.929	0.802	0.747	0.668
Gram + monomial moments	0.869	0.701	0.938	0.812	0.802	0.705
monomial moments only	0.891	0.783	0.939	0.873	0.904	0.887

alone, a doubled training schedule, etc. Each changed the result by at most 0.02, and not consistently in the same direction across strata.

Two further checks suggest the limit is in the release and not in the network. First, on tables drawn from the model's own prior, the noise-free model also sits at ridge level. Second, a linear read-out of the release, fit without privacy on the training rows themselves, does no better. Our interpretation is that a one-dimensional slice describes the target along a single projection, so any reader of such slices is limited to roughly a sum of smooth ridge functions, i.e., a linear model with a little curvature. Moving to moments over pairs of projections was the only change that lowered the floor, to 0.542 and 0.744, and this is the release used by the model we report.

G.6 A CURRICULUM EFFECT

The large suite showed something we did not expect. With 50k rows and $\epsilon = 8$, the release carries a fifth of the noise it carries at 10k rows, yet the private model improves by only 0.013. We first checked whether the network was misreading the size and noise tokens, by clamping them to their training range, but it is not the cause. Instead, we find that the privately trained model scores 0.683 on the 50k-row tables even when handed a noise-free release, against 0.627 for the model trained without noise, shown in Figure 14. The second stage of the curriculum (rows up to 10k, budgets up to 10) never contains a release without noise.

Mixing 30 percent clean releases into the curriculum brings the clean-release score to 0.635 and the private score at $\epsilon = 8$ from 0.696 to 0.677 (the DP-SGD MLP is at 0.649), at a cost of at most 0.03 on the small tables. We did not include this change in the reported model; every number in the main text is for the private curriculum. We also tried widening the budget range of the curriculum to

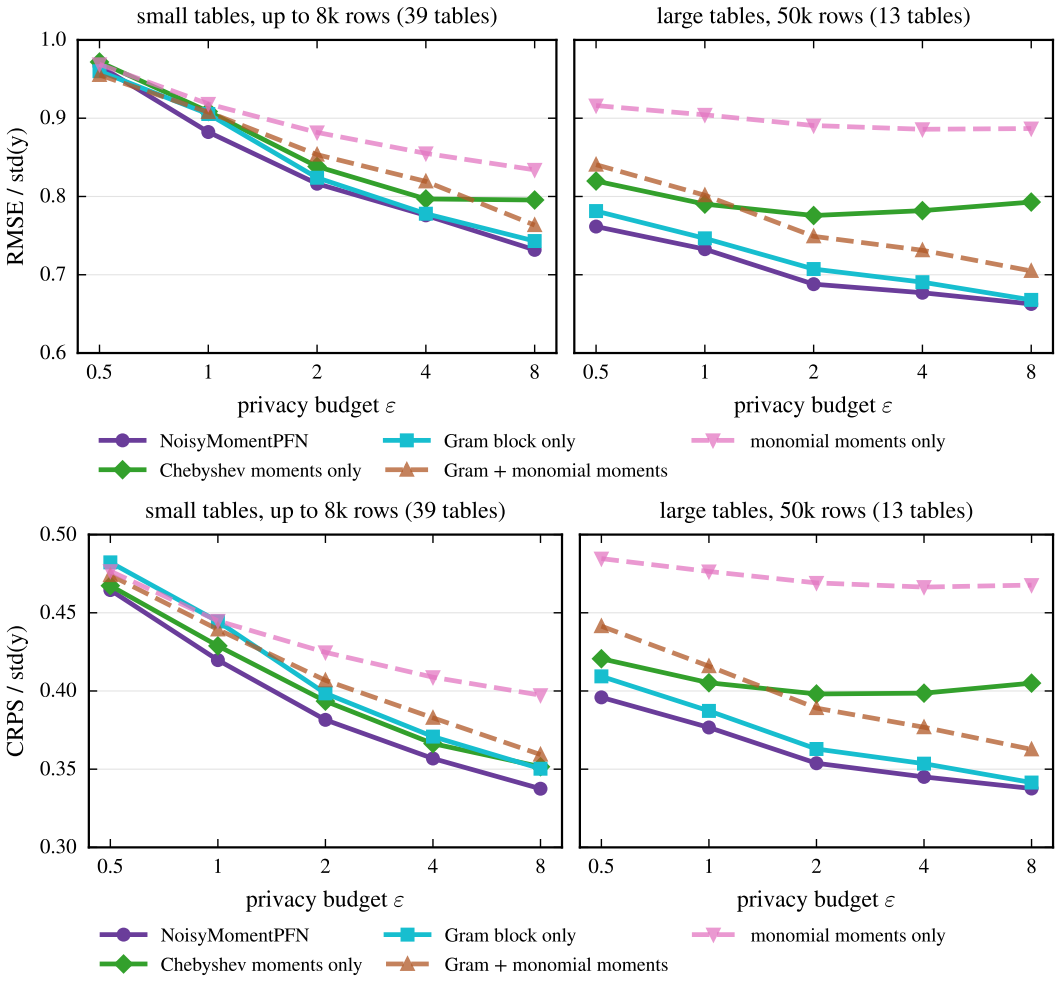


Figure 12: Release ablations: normalized RMSE (top) and CRPS (bottom) against the privacy budget on the 39 small tables (left) and the 13 large tables at 50k rows (right), for the five releases described in Section G.4.

Table 4: Release ablations of NoisyMomentPFN: mean normalized CRPS at $\varepsilon = 1$ and $\varepsilon = 8$ on each suite (tables per suite: numeric 17, mixed 22, large, 50k rows 13). Every model shares the architecture, prior, noisy curriculum and 60k step schedule; only the released statistics differ. Best per column in bold.

	numeric		mixed		large, 50k rows	
release	$\varepsilon = 1$	$\varepsilon = 8$	$\varepsilon = 1$	$\varepsilon = 8$	$\varepsilon = 1$	$\varepsilon = 8$
Gram + Chebyshev moments (NoisyMomentPFN)	**0.406**	**0.308**	**0.430**	**0.360**	**0.377**	**0.338**
Chebyshev moments only	0.414	0.321	0.440	0.375	0.405	0.405
Gram block only	0.446	0.327	0.443	0.368	0.387	0.341
Gram + monomial moments	0.437	0.342	0.441	0.373	0.416	0.363
monomial moments only	0.444	0.385	0.446	0.407	0.476	0.468

300, so that low-noise releases occur throughout training; this made the model worse at every noise level, since the budgets the benchmark actually uses are then seen far less often. We conclude that a private model should be trained on the noise levels it will see at inference, including the nearly clean releases that large tables produce at large budgets, and that initialization from a clean model is not

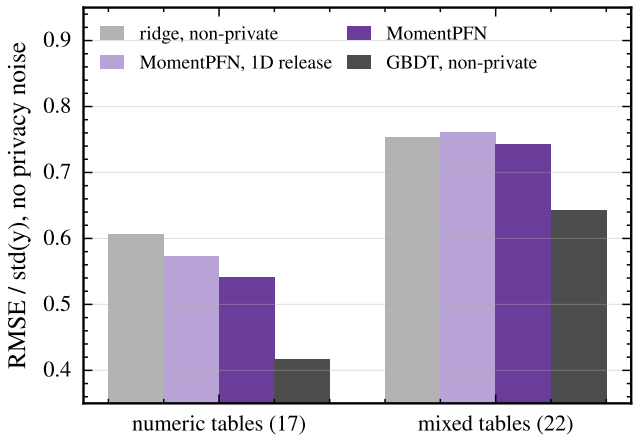


Figure 13: The noise-free floor of the representation: normalized RMSE of MomentPFN trained and scored without privacy noise with one-dimensional slices (256 directions) and with pairs (128 pairs), between non-private ridge and gradient boosting, on the numeric and mixed strata.

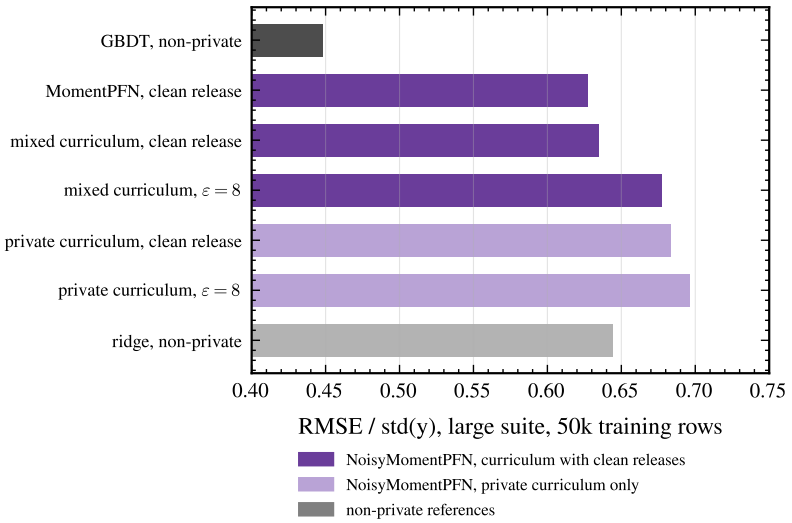
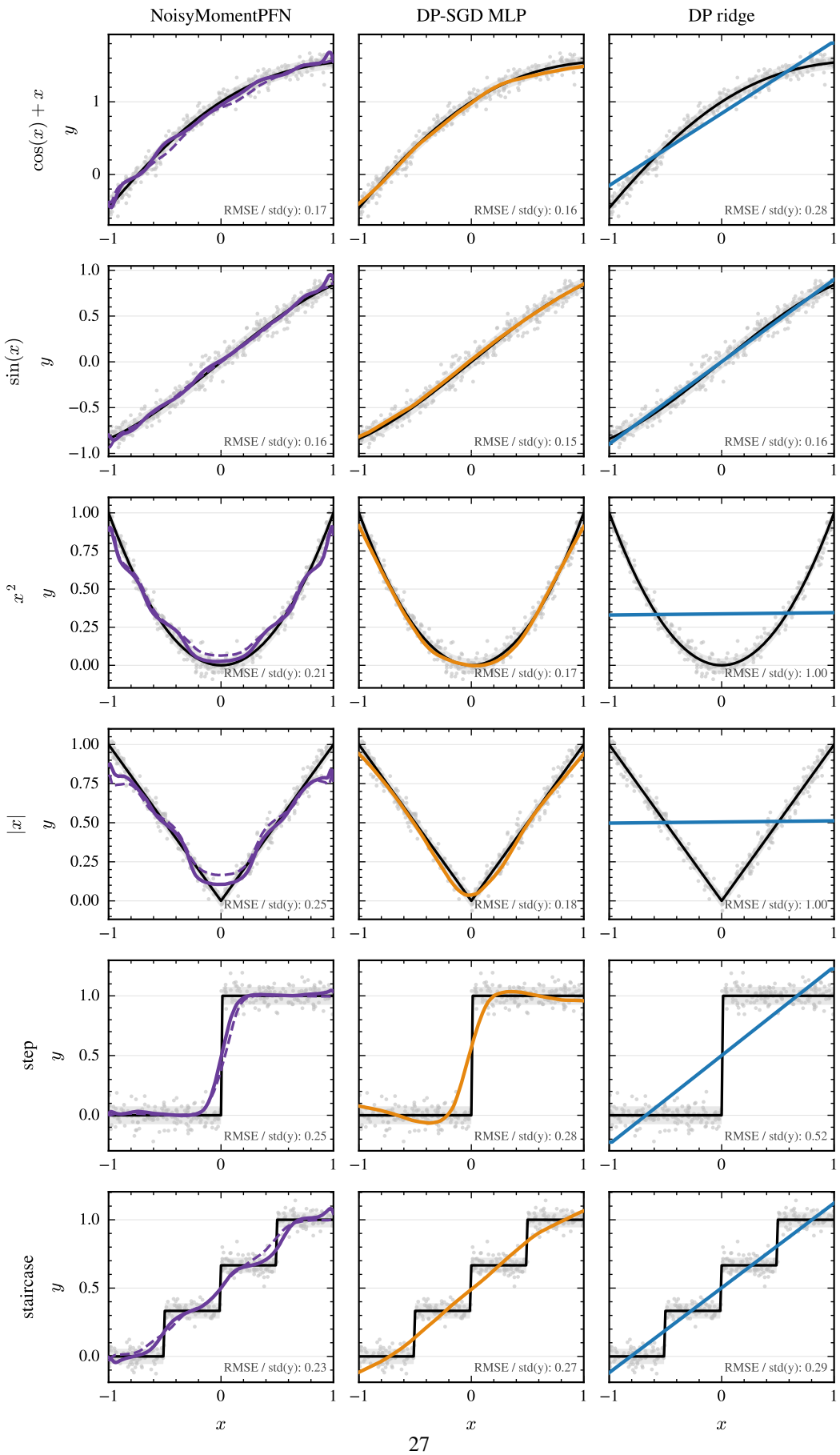


Figure 14: The curriculum effect on the large suite with 50k training rows, normalized RMSE. The privately trained model (light) scores far above the model trained without noise even when given a noise-free release; mixing 30 percent clean releases into its curriculum (dark) recovers most of the difference, on clean releases and at $\epsilon = 8$ alike.

a substitute for keeping clean releases in the mix. This also suggests that the gap on small tables at small budgets is an information limit, while the gap on large tables was a limit of our curriculum.

G.7 One-dimensional examples

Figure 15 extends the staircase example of Figure 4 to six functions of one variable at the small budget $\epsilon = 1$, with the same recipe: $10,000$ training rows drawn uniformly on $[-1, 1]$, homoscedastic noise at 0.15 of the function's standard deviation, public bounds at the range of the target plus three noise standard deviations, one data and one privacy draw. At this budget the release is noisy enough that NoisyMomentPFN rounds every sharp feature, the step and the staircase most visibly, while the smooth functions are recovered at the level of the DP-SGD MLP, while DP ridge fits a line in every panel as expected.


NoisyMomentPFN
DP-SGD MLP
DP ridge
cos(x) + x
sin(x)
x²
|x|
step
staircase
y
x
RMSE / std(y): 0.17
RMSE / std(y): 0.16
RMSE / std(y): 0.28
RMSE / std(y): 0.16
RMSE / std(y): 0.15
RMSE / std(y): 0.16
RMSE / std(y): 0.21
RMSE / std(y): 0.17
RMSE / std(y): 1.00
RMSE / std(y): 0.25
RMSE / std(y): 0.18
RMSE / std(y): 1.00
RMSE / std(y): 0.25
RMSE / std(y): 0.28
RMSE / std(y): 0.52
RMSE / std(y): 0.23
RMSE / std(y): 0.27
RMSE / std(y): 0.29
training rows
truth, one noise std
truth
MomentPFN
NoisyMomentPFN
DP-SGD MLP
DP ridge