CONTINUOUS DIFFUSION FOR MIXED-TYPE TABULAR DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Score-based generative models (or diffusion models for short) have proven successful for generating text and image data. However, the adaption of this model family to tabular data of mixed-type has fallen short so far. In this paper, we propose CDTD, a Continuous Diffusion model for mixed-type Tabular Data. Specifically, we combine score matching and score interpolation to ensure a common continuous noise distribution for *both* continuous and categorical features alike. We counteract the high heterogeneity inherent to data of mixed-type with distinct, adaptive noise schedules per feature or per data type. The learnable noise schedules ensure optimally allocated model capacity and balanced generative capability. We homogenize the data types further with model-specific loss calibration and initialization schemes tailored to mixed-type tabular data. Our experimental results show that CDTD consistently outperforms state-of-the-art benchmark models, captures feature correlations exceptionally well, and that heterogeneity in the noise schedule design boosts the sample quality.

1 INTRODUCTION

Score-based generative models (Song et al., 2021), also termed diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), have shown remarkable potential for the generation of images (Dhariwal & Nichol, 2021; Rombach et al., 2022), videos (Ho et al., 2022), text (Li et al., 2022; Dieleman et al., 2022; Wu et al., 2023), molecules (Hoogeboom et al., 2022), and many other highly complex data structures with continuous features. The framework has since been adapted to categorical data in various ways, including discrete diffusion processes (Austin et al., 2021; Hoogeboom et al., 2021), diffusion in continuous embedding space (Dieleman et al., 2022; Li et al., 2022; Regol & Coates, 2023; Strudel et al., 2022), and others (Campbell et al., 2022; Meng et al., 2022; Sun et al., 2023). Diffusion models which include both, continuous and categorical features alike, build directly on advances from the image domain (Kim et al., 2023; Kotelnikov et al., 2023; Lee et al., 2023; Jolicoeur-Martineau et al., 2024) and thus, are not designed to deal with challenges specific to mixed-type tabular data: The different diffusion processes and their losses are neither aligned nor balanced across data types, and do not scale to larger datasets and/or features with a greater number of categories. Models that naively combine different losses to integrate distinct generative processes may suffer from implicitly favoring the sample quality of some features or data types over others (Ma et al., 2020). Previously proposed diffusion models for tabular data (e.g., Kotelnikov et al., 2023; Lee et al., 2023), often use a discrete diffusion framework to model categorical features. However, this fails to capture the full uncertainty during the denoising process, as a data sample can never be 'in-between' categories at any point in the reverse process.

A crucial component in score-based generative models is the noise schedule (Kingma et al., 2022; Chen et al., 2022; Chen, 2023; Jabri et al., 2022; Wu et al., 2023). Typical noise schedules for image and text data are designed to focus model capacity on the noise levels most important to sample quality (Nichol & Dhariwal, 2021; Karras et al., 2022), while others attempt to learn the optimal noise schedule (Dieleman et al., 2022; Kingma et al., 2022). For mixed-type tabular data, existing approaches often combine distinct diffusion processes for the continuous and discrete features to derive a joint model (Kotelnikov et al., 2023; Lee et al., 2023). However, noise schedules are not directly transferable from one data modality to another and therefore, using specifications from image or text domain models is not optimal: First, the inherently different diffusion processes make it difficult to balance the noise schedules across features and feature types, and negatively affect the allocation of model capacity across timesteps. For instance, both TabDDPM (Kotelnikov et al., 2023) and CoDi (Lee et al., 2023) use the discrete multinomial diffusion framework (Hoogeboom et al., 2021) to model categorical features. This induces different types of noise for continuous and categorical features, making an alignment or even comparison of noise schedules impossible. Second, and most importantly, the domain, nature and marginal distribution can vary significantly across features (Xu et al., 2019). For instance, any two continuous features may be subject to different levels of discretization or different bounds, even after applying common data pre-processing techniques; and any two categorical features may differ in the number of categories, or the degree of imbalance. The high heterogeneity and lack of balancing warrants a rethinking of fundamental parts of the diffusion framework, including the noise schedule and the effective combination of diffusion processes for different data types.

In this paper, we introduce *Continuous Diffusion for mixed-type Tabular Data* (CDTD) to address the aforementioned shortcomings. We combine *score matching* (Hyvärinen, 2005) with *score interpolation* (Dieleman et al., 2022) to derive a score-based model that pushes the diffusion process for categorical data into embedding space, and uses a Gaussian diffusion process for *both* continuous and categorical features. This way, the different noise processes become directly comparable, easier to balance, and enable the application of, for instance, classifier-free guidance (Ho & Salimans, 2022), accelerated sampling (Lu et al., 2022), and other advances, to mixed-type tabular data.

We counteract the high feature heterogeneity inherent to data of mixed-type with distinct feature or type-specific adaptive noise schedules. The learnable noise schedules allow the model to directly take feature or type heterogeneity into account during both training and generation, and thus avoid the reliance on image or text-specific noise schedule designs. Moreover, we propose a diffusion-specific loss normalization and initialization scheme to homogenize different data types and their losses effectively. Our improvements ensure a better allocation of the model's capacity across features, feature types and timesteps, and yield high quality samples of tabular data. CDTD outperforms state-of-the-art baseline models across a diverse set of sample quality metrics as well as computation time for data sets with an arbitrary number of categories and data points. Our experiments show that CDTD captures feature correlations exceptionally well, and that explicitly allowing for data-type heterogeneity in the noise schedules benefits sample quality.

In sum, we make several contributions specific to diffusion probabilistic modeling of tabular data:

- We propose a joint continuous diffusion model for *both* continuous and categorical features such that all noise distributions are Gaussian.
- We balance model capacity across continuous and categorical features with a novel and effective loss calibration, an adjusted score model initialization and type or feature-specific noise schedules.
- We extend the idea of timewarping and propose a functional form to efficiently learn adaptive noise schedules, and to allow for exact evaluation and easy incorporation of prior information on the relative importance of noise levels.
- We drastically improve the scalability of tabular data diffusion models to features with a high number of categories.
- We boost the quality of the generated samples with adaptive, feature or type-specific noise schedules.
- Our CDTD model allows the first-ever use of advanced techniques, like classifier-free guidance, for mixed-type tabular data directly in data space.

2 Score-based Generative Framework

We start with a brief outline of the score-based frameworks for continuous and categorical features. Next, we combine these into a single diffusion model to learn the joint distribution of mixed-type data.

2.1 Continuous Features

We denote $x_{\text{cont}}^{(i)} \in \mathbb{R}$ as the i-th continuous feature and $\mathbf{x}_0 \equiv \mathbf{x}_{\text{cont}} \in \mathbb{R}^{K_{\text{cont}}}$ as the stacked feature vector. Further, let $\{\mathbf{x}_t\}_{t=0}^{t=1}$ be a diffusion process that gradually adds noise in continuous time $t \in [0, 1]$ to $\mathbf{x}_0$, and let $p_t(\mathbf{x})$ denote the density function of the data at time t. Then, this process

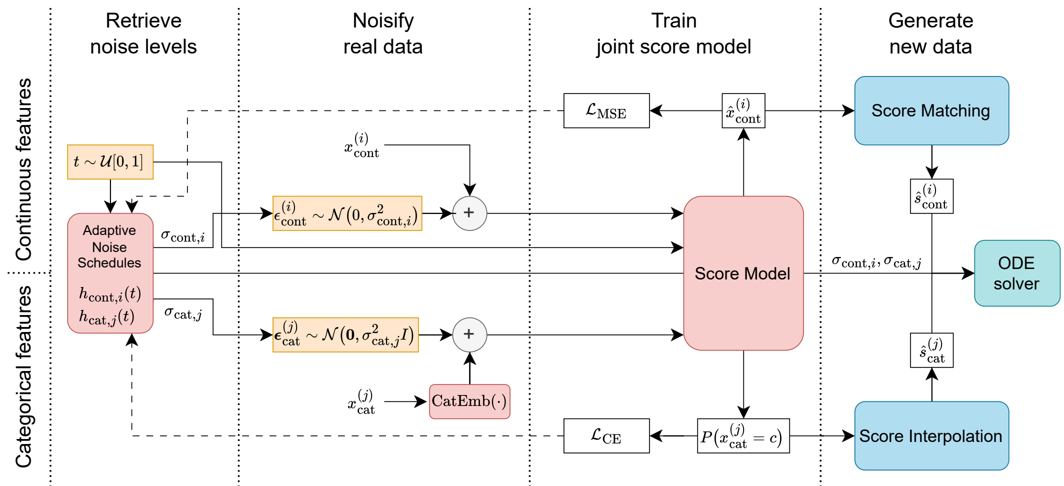

Figure 1: CDTD framework. Adaptive noise schedules are trained to fit the (possibly aggregated) MSE and CE losses and transform the uniform timestep t to a potentially feature-specific noise level to diffuse ("noisify") the scalar values (for continuous features) or the embeddings (for categorical features). Associated sampling processes are highlighted in orange. The approximated score functions are concatenated and passed to an ODE solver for sample generation.

transforms the real data distribution $p_0(\mathbf{x})$ into a terminal distribution of pure noise $p_1(\mathbf{x})$ from which we can sample. Our goal is to learn the reverse process that allows us to go from noise $\mathbf{x}_1 \sim p_1(\mathbf{x})$ to a new data sample $\mathbf{x}_0^* \sim p_0(\mathbf{x})$.

The forward-pass of this continuous-time diffusion process is formulated as the solution to a stochastic differential equation (SDE):

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{1}$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^{K_{\text{cont}}} \to \mathbb{R}^{K_{\text{cont}}}$ is the drift coefficient, $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient, and $\mathbf{w}$ is a Brownian motion (Song et al., 2021). The reversion yields the trajectory of $\mathbf{x}$ as t goes backwards in time from 1 to 0, and is formulated as a probability flow ordinary differential equation (ODE):

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t. \tag{2}$$

We approximate the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, the only unknown in Equation (2), by training a time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ via *score matching* (Hyvärinen, 2005). The parameters $\boldsymbol{\theta}$ are trained to minimize the *denoising score matching* objective:

$$\mathbb{E}_t \left[\lambda_t \mathbb{E}_{\mathbf{x}_0} \mathbb{E}_{\mathbf{x}_t|\mathbf{x}_0} \left\| s_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t|\mathbf{x}_0) \right\|_2^2\right], \tag{3}$$

where $\lambda_t : [0, 1] \to \mathbb{R}_+$ is a positive weighting function for timesteps $t \sim \mathcal{U}_{[0,1]}$, and $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ is the density of the noisy $\mathbf{x}_t$ given the ground-truth data $\mathbf{x}_0$ (Vincent, 2011).

In this paper, we use the EDM formulation (Karras et al., 2022), that is, $\mathbf{f}(\cdot, t) = \mathbf{0}$ and $g(t) = \sqrt{2[\frac{d}{dt}\sigma(t)]\sigma(t)}$ such that $p_{0t}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_0, \sigma^2(t) I_{K_{\text{cont}}})$. We start the reverse process with sampling $\mathbf{x}_1 \sim p_1(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma^2(1) I_{K_{\text{cont}}})$ for $\sigma^2(1)$ being sufficiently large and $\mathbb{E}[\mathbf{x}_0] = \mathbf{0}$. We then gradually guide $\mathbf{x}_1$ towards high density regions in the data space with $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ replacing the unknown, true score function in Equation (2). In practice, ODE or predictor-corrector samplers can be used for this iterative denoising process (Song et al., 2021).

2.2 Categorical Features

Let $x_{\text{cat}}^{(j)}$ denote a single observation of the j-th categorical feature which can take on any of C_j possible classes $c \in \{1, \ldots, C_j\}$. We learn a feature-specific encoder to represent each category c

as a d-dimensional vector $\mathbf{e}_c^{(j)} = \mathrm{Enc}_j(x_{\text{cat}}^{(j)})$. Further, let $\mathbf{x}_0^{(j)} \in \{\mathbf{e}_1^{(j)}, \dots, \mathbf{e}_{C_j}^{(j)}\}$ be the noiseless embedding at $t=0$ (to highlight $\mathbf{e}_c^{(j)}$ as the ground-truth in the diffusion framework). To maximize the integrability of the diffusion frameworks for categorical and continuous data, we impose the same Gaussian-type noise on categorical *and* continuous features. We thus produce a noisy embedding $\mathbf{x}_t^{(j)} \sim p_{0t}(\mathbf{x}_t^{(j)}|\mathbf{x}_0^{(j)}) = \mathcal{N}(\mathbf{x}_t^{(j)}|\mathbf{x}_0^{(j)}, \sigma^2(t)I_d)$ such that $\mathbf{x}_1^{(j)} \sim p_1(\mathbf{x}^{(j)}) = \mathcal{N}(\mathbf{0}, \sigma^2(1)I_d)$, analogous to score matching.

For categorical data, denoising score matching (see Equation (3)) is not directly applicable to training a score model to learn $\nabla_{\mathbf{x}_t^{(j)}} \log p_{0t}(\mathbf{x}_t^{(j)}|\mathbf{x}_0^{(j)})$, since the score can only take on C_j distinct values. To proceed, we transform the score matching approach into a discrete choice problem. Note that for a given t and $\mathbf{x}_t^{(j)}$ it is sufficient to find $\mathbb{E}_{p(\mathbf{x}_0^{(j)}|\mathbf{x}_t^{(j)},t)}[\nabla_{\mathbf{x}_t^{(j)}} \log p_{0t}(\mathbf{x}_t^{(j)}|\mathbf{x}_0^{(j)})]$ as it minimizes Equation (3). Assuming Gaussian noise, we have

$$\mathbb{E}_{p(\mathbf{x}_0^{(j)}|\mathbf{x}_t^{(j)},t)}\left[\nabla_{\mathbf{x}_t^{(j)}} \log p_{0t}(\mathbf{x}_t^{(j)}|\mathbf{x}_0^{(j)})\right] = \frac{1}{\sigma^2(t)}\left[\mathbb{E}_{p(\mathbf{x}_0^{(j)}|\mathbf{x}_t^{(j)},t)}[\mathbf{x}_0^{(j)}] - \mathbf{x}_t^{(j)}\right]. \tag{4}$$

We can thus approximate the score by computing $\hat{\mathbf{x}}_0^{(j)} = \mathbb{E}_{p(\mathbf{x}_0^{(j)}|\mathbf{x}_t^{(j)},t)}[\mathbf{x}_0^{(j)}]$, i.e., a probability weighted average of the C_j possible embedding vectors. Since $p(\mathbf{x}_0^{(j)} = \mathbf{e}_c^{(j)}|\mathbf{x}_t^{(j)}, t) = p(x_{\text{cat}}^{(j)} = c|\mathbf{x}_t^{(j)}, t)$, we can estimate $p(\mathbf{x}_0^{(j)}|\mathbf{x}_t^{(j)}, t)$ via a classifier that predicts the C_j class probabilities and is trained to minimize the cross-entropy (CE). This procedure interpolates between the C_j ground-truth embeddings $\mathbf{x}_0^{(j)}$ and is therefore known as *score interpolation* (Dieleman et al., 2022).

This framework can easily be extended to multiple categorical features. Most importantly, Enc_j is trained alongside the model such that $\mathbf{x}_0^{(j)}$ is directly optimized for denoising the data. Since the reverse process also happens in embedding space, the model only has to commit to a category at the final step of generation, i.e., we allow for a smooth, continuous transition between states at intermediate timesteps. This is unlike multinomial diffusion (Hoogeboom et al., 2021), which models categorical data based on *discrete* transitioning steps. By defining diffusion for categorical data in embedding space, we allow our model to fully take uncertainty at intermediate timesteps into account, which improves the consistency of the generated samples (Dieleman et al., 2022). Therefore, the adaption of score interpolation allows CDTD to capture subtle dependencies both *within* and *across* data types more accurately.

3 METHOD

In short, we combine *score matching* (Equation (3)) with *score interpolation* (Equation (4)) to model the joint distribution of mixed-type data. Next, we discuss the important components of our method. In particular, the combination of the different losses for score matching and score interpolation, initialization and loss weighting concerns, and the adaptive type- or feature-specific noise schedule designs.

3.1 GENERAL FRAMEWORK

Figure 1 gives an overview of our Continuous Diffusion for mixed-type Tabular Data (CDTD) framework. The score model is conditioned on (1) all noisy continuous features, (2) the noisy embeddings of all categorical features in Euclidean space, and (3) the timestep t which reflects potentially feature-specific, adaptive noise levels $\sigma_{\text{cont},i}$ and $\sigma_{\text{cat},j}$ for all i and j. Additional conditioning information, such as the target feature for classification tasks, are straightforward to add. Note that while the Gaussian noise process acts directly on the continuous features, it acts on the *embeddings* of the categorical features. This way, we ensure a common continuous noise process for both data types.

During training, the model predicts the ground-truth value for continuous features and the class-specific probabilities for categorical features. During generation, we concatenate the score estimates, $\hat{s}_{\text{cont}}^{(i)}$ and $\hat{s}_{\text{cat}}^{(j)}$, for all features i and j, and pass them to an ODE solver together with $\sigma_{\text{cont},i}$ and $\sigma_{\text{cat},j}$, the noise levels retrieved by transforming linearly spaced timesteps with the learned adaptive noise schedules. Further details on the implementation and sampling are provided in Appendix J and Appendix K, respectively.

3.2 HOMOGENIZATION OF DATA TYPES

Let $\mathcal{L}_{\text{MSE}}(x_{\text{cont}}^{(i)}, t)$ denote the time-weighted MSE (i.e., score matching) loss of the i-th continuous feature at a single timestep t, and $\mathcal{L}_{\text{CE}}(x_{\text{cat}}^{(j)}, t)$ the CE (i.e., score interpolation) loss of the j-th categorical feature. Naturally, the two losses are defined on different scales. This leads to an unintended importance weighting of features in the generative process (Ma et al., 2020). We assume that an unconditional model should a priori, i.e., without having any information, be indifferent between all features. This reflects the state of the model at the terminal timestep $t = 1$ in the diffusion process.

Formally, we aim to find calibrated losses, $\mathcal{L}_{\text{MSE}}^*$ and $\mathcal{L}_{\text{CE}}^*$ for all continuous features i and categorical features j, such that

$$\mathbb{E}[\mathcal{L}_{\text{MSE}}^*(x_{\text{cont}}^{(i)}, 1)] = \mathbb{E}[\mathcal{L}_{\text{CE}}^*(x_{\text{cat}}^{(j)}, 1)] = 1. \tag{5}$$

For continuous features, $\mathbb{E}[\mathcal{L}_{\text{MSE}}^*(x_{\text{cont}}^{(i)}, 1)] = 1$ follows from standardizing $x_{\text{cont}}^{(i)}$ to zero mean and unit variance. For categorical features, we compute the normalization constant $\mathbb{E}[\mathcal{L}_{\text{CE}}(x_{\text{cat}}^{(j)}, 1)]$ directly as the CE of each predicted class in proportion to its empirical distribution in the train set (see Appendix A). We then average the calibrated losses to derive the joint loss function at a given timestep:

$$\mathcal{L}(t) = \frac{1}{K}\Big[\sum_{i=1}^{K_{\text{cont}}} \mathcal{L}_{\text{MSE}}^*(x_{\text{cont}}^{(i)}, t) + \sum_{j=1}^{K_{\text{cat}}} \mathcal{L}_{\text{CE}}^*(x_{\text{cat}}^{(j)}, t)\Big], \tag{6}$$

where $K = K_{\text{cont}} + K_{\text{cat}}$.

The loss calibration and the multiple data modalities have implications for the optimal initialization of the score model. We aim to initialize all *feature-specific* losses at one. We therefore initialize the output layer weights to zero (like in image diffusion models) and the output biases for continuous features to zero, and rely on the timestep weights of the EDM parameterization (Karras et al., 2022) to achieve a unit loss for all t. For the categorical features, we initialize the biases to match the category's empirical probability in the training set (see Appendix B).

The initial equal importance across all timesteps will naturally change over the course of training. We employ a normalization scheme for the average diffusion loss (Karras et al., 2023; Kingma & Gao, 2023) to allow for changes in relative importance among features but ensure equal importance of all timesteps throughout training. To do so, we learn the time-specific normalization term $Z(t)$ such that $\mathcal{L}(t)/Z(t) \approx 1$. This ensures a consistent gradient signal and can be implemented by training a neural network to predict $\mathcal{L}(t)$ alongside our diffusion model (for details see Appendix C).

3.3 NOISE SCHEDULES

Since the optimal noise schedule of one feature impacts the noise schedules of other features, and different data types have different sensitivities to additive noise, we introduce *feature-specific* or *type-specific* noise schedules. For instance, given the same embedding dimension, more noise is needed to remove the same amount of signal from embeddings of features with fewer classes. Likewise, a delayed noise schedule for one feature might improve sample quality as the model can rely on other correlated features that have been (partially) generated first. We make the noise schedules learnable, and therewith *adaptive* to avoid the reliance on designs for other data modalities.

We investigate the following noise schedule variants: (1) a single adaptive noise schedule, (2) adaptive noise schedules differentiated per data type and (3) feature-specific adaptive noise schedules. We only introduce the feature-specific noise schedules explicitly. The other noise schedule types are easily derived from our argument by appropriately aggregating terms across features.

Feature-specific Noise Schedules. According to Equation (1), and following the EDM parameterization (Karras et al., 2022), we define the diffusion process of the i-th continuous feature as

$$\mathrm{d}x_{\text{cont}}^{(i)} = \sqrt{2\Big[\frac{d}{dt}h_{\text{cont},i}(t)\Big]h_{\text{cont},i}(t)}\mathrm{d}w_t^{(i)}, \tag{7}$$

and likewise the trajectory of the j-th categorical feature as

$$\mathrm{d}\mathbf{x}_{\text{cat}}^{(j)} = \sqrt{2\Big[\frac{d}{dt}h_{\text{cat},j}(t)\Big]h_{\text{cat},j}(t)}\mathrm{d}\mathbf{w}_t^{(j)}, \tag{8}$$

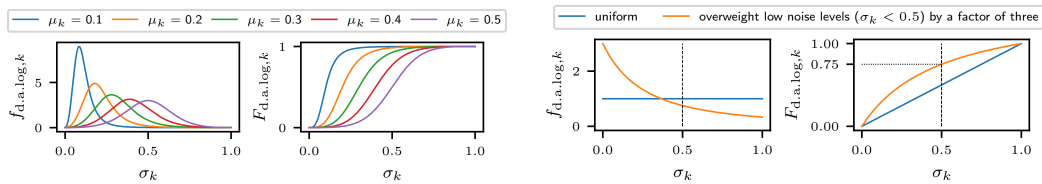

Figure 2: (Left) pdf ($f_{\mathrm{d.a.log},k}$) and cdf ($F_{\mathrm{d.a.log},k}$) of the domain-adapted Logistic distribution for five different values of the location parameter μ_k and for a given curve steepness $\nu_k = 3$. (Right) impact of uniform vs. adjusted timewarping initialization on the pdf ($f_{\mathrm{d.a.log},k}$) and the cdf ($F_{\mathrm{d.a.log},k}$).

where $\mathbf{x}_{\text{cat}}^{(j)}$ is the d-dimensional embedding of $x_{\text{cat}}^{(j)}$ in Euclidean space. The *feature-specific* noise schedules $h_{\text{cont},i}(t)$ and $h_{\text{cat},j}(t)$ represent the standard deviations of the added Gaussian noise such that $\sigma_{\text{cont},i}(t) = h_{\text{cont},i}(t)$ and $\sigma_{\text{cat},j}(t) = h_{\text{cat},j}(t)$. Thus, each continuous feature and each embedded categorical feature is affected by a distinct noise schedule.

Adaptive Noise Schedules. Based on Dieleman et al. (2022), we aim to learn a noise schedule $h_k : t \mapsto \sigma$ for all $K = K_{\text{cont}} + K_{\text{cat}}$ features. Note that $t \in [0, 1]$, and with pre-specified minimum and maximum noise levels, we can scale σ_k to lie in $[0, 1]$ as well, without loss of generality. We will learn the feature-specific loss given the noise level, $F_k : \sigma_k \mapsto \ell_k$, alongside the score model, with ℓ_k the relevant (not explicitly weighted) training loss for the k-th feature. Then, our mapping of interest is $h_k = \tilde{F}_k^{-1}$, that is, the normalized and inverted function F_k. This encourages the relation between t and ℓ_k to be linear.

Higher noise levels imply a lower signal-to-noise ratio, and therefore a larger incurred loss for the score model. Accordingly, F_k must be a monotonically increasing and *S-shaped* function. We let $F_k = \gamma_k F_{\mathrm{d.a.log},k}(\sigma_k)$ where $\gamma_k > 0$ is a scaling factor that at $t = 1$ enables fitting a loss $\ell_k > 1$ early on in the training process, and a loss $\ell_k < 1$ in case conditioning information is included. Further, we use the cdf of the domain-adapted Logistic distribution $F_{\mathrm{d.a.log},k}(\sigma_k)$, where the input is pre-processed via a Logit function, with parameters $0 < \mu_k < 1$ (the location of the inflection point) and $\nu_k \geq 1$ (the steepness of the curve). Figure 2 illustrates the effect of the location parameter. The implicit importance of the noise levels is conveniently represented by the corresponding pdf $f_{\mathrm{d.a.log},k}$. To normalize and invert F_k, we set $\gamma_k = 1$ and and directly utilize the quantile function $F_{\mathrm{d.a.log},k}^{-1}$. The detailed derivation of all relevant functions is given in Appendix D.

Our functional choice has several advantages. First, each noise schedule can be evaluated exactly without the need for approximations and only requires three parameters. Second, these parameters are well interpretable in the diffusion context and provide information on the inner workings of the model. For instance, for $\mu_1 < \mu_2$, the model starts generating feature 2 before feature 1 in the reverse process. Third, the proposed functional form is less flexible than the original piece-wise linear function (Dieleman et al., 2022) such that an exponential moving average on the parameters is not necessary, and the fit is more robust to "outliers" encountered during training.

We use the adaptive noise schedules during both training and generation. We derive importance weights from $f_{\mathrm{d.a.log},k}$ to fit h_k to avoid biasing the noise schedule to timesteps that are frequently sampled during training. Type-specific noise schedules refer to learning two functions F_1 and F_2 that predict the respective average loss over all features of a data type. Examples of learned noise schedules are given in Appendix O.

3.4 Additional Customization to Tabular Data

In the diffusion process, we add noise directly to the continuous features but to the embeddings of categorical features. We generally need more noise to remove all signal from the categorical representations. We therefore define *type-specific* minimum and maximum noise levels: For categorical features, we let $\sigma_{\text{cat},\min} = 0.1$ and $\sigma_{\text{cat},\max} = 100$; for continuous features, we set $\sigma_{\text{cont},\min} = 0.002$ and $\sigma_{\text{cont},\max} = 80$ (see Karras et al., 2022).

Lastly, an uninformative initialization of the adaptive noise schedules requires to set $\mu_k = 0.5, \nu_k \approx 1$ and $\gamma_k = 1$ such that $F_{\mathrm{d.a.log},k}$ corresponds approximately to the cdf of a uniform distribution. We can improve this with a more informative prior: In the image domain, diffusion models allocate substantial capacity towards generating the high level structure before generating details at lower noise levels. In tabular data, the location of features in the data matrix, and therefore the high level structure, is fixed. Instead, we are interested in generating details as accurately as possible, as these influence, for instance, subtle correlations among features. Note that the inflection point, μ_k, of our adaptive noise schedule corresponds to the proportion of high (normalized) noise levels (i.e., $\sigma_k \geq 0.5$) in the distribution. Therefore, we adjust the initial noise schedules such that low noise levels ($\sigma_k < 0.5$) are weighted by a factor of 3 relative to high noise levels ($\sigma_k \geq 0.5$) (see Figure 2). The proportion of high noise levels is decreased to $\mu_k = 1/4$. We let $\nu_k \approx 1$ for a dispersed initial probability mass and initialize the scaling factor to $\gamma_k = 1$.

4 EXPERIMENTS

We benchmark our model against several generative models across multiple datasets. Additionally, we investigate three different noise schedule specifications: (1) a single adaptive noise schedule for both data types (*single*), (2) continuous and categorical data type-specific adaptive noise schedules (*per type*), and (3) feature-specific adaptive noise schedules (*per feature*).

Baseline models. We use a diverse benchmark set of state-of-the-art generative models for mixed-type tabular data. This includes SMOTE (Chawla et al., 2002), ARF (Watson et al., 2023), CTGAN (Xu et al., 2019), TVAE (Xu et al., 2019), TabDDPM (Kotelnikov et al., 2023), CoDi (Lee et al., 2023), TabSyn (Zhang et al., 2024). Each model follows a different design and/or modeling philosophy. Note that CoDi is an extension of STaSy (Kim et al., 2023, the same group of authors) that has shown to be superior in performance. For scaling reasons, ForestDiffusion (Jolicoeur-Martineau et al., 2024) is not an applicable benchmark.[1] Further details on the respective benchmark models and their implementations are provided in Appendix F and Appendix G. We provide an in-depth comparison of CDTD to the diffusion-based baselines in Appendix N. To keep the comparison fair, we use the same architecture for CDTD as TabDDPM (the latter has also been adopted by TabSyn), with minor changes to accommodate the different inputs (see Appendix J).

Datasets. We systematically investigate our model on eleven publicly available datasets. The datasets vary in size, prediction task (regression vs. binary classification[2]), number of continuous *and* categorical features and their distributions. The number of categories for categorical features varies significantly across datasets (for more details, see Appendix E). We remove observations with missings in the target or any of the continuous features and encode missings in the categorical features as a separate category. All datasets are split in train (60%), validation (20%) and test (20%) partitions, hereinafter denoted $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$, respectively. For classification tasks, we use stratification with respect to the outcome, we condition the model on the outcome during training and generation, and use the train set proportions for generation. In a last post-processing step, we round the integer-valued continuous features after generation for all models.

4.1 EVALUATION METRICS

In our experiments, we follow conventions from previous papers and use four sample quality criteria, which we assess using a comprehensive set of measures. All metrics are averaged over five random seeds that affect the generative process, which samples synthetic data $\mathcal{D}_{\text{gen}}$ of size $\min(|\mathcal{D}_{\text{train}}|, 50\,000)$.

Machine learning efficiency. We follow the conventional train-synthetic-test-real strategy (see, Borisov et al., 2023; Liu et al., 2023; Kotelnikov et al., 2023; Kim et al., 2023; Xu et al., 2019; Watson et al., 2023). Hence, we train a group of models, consisting of a (logistic/ridge) regression, a

[1] Jolicoeur-Martineau et al. (2024) report in the appendix that they used 10-20 CPUs with 64-256 GB of memory for datasets with a median number of 540 observations. With the suggested hyperparameters (for improved efficiency) and 64 CPUs, the model took approx. 500 min of training on the relatively small `nmes` data. Note that the model estimates KT separate models, with K being the number of features and T the noise levels. Therefore, we consider ForestDiffusion to be prohibitively expensive for higher-dimensional data generation.

[2] For ease of presentation, we only analyze binary targets. However, CDTD trivially extends to targets with multiple classes.

Table 1: Average performance rank of each generative model across eleven datasets. Per metric, **bold** indicates the best, underline the second best result. We assigned the rank 10 for CoDi on `lending` and `diabetes`, TabDDPM on `acsincome` and `diabetes`, SMOTE on `acsincome` and `covertype`.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
RMSE	$\underline{3.400}_{\pm 3.382}$	$3.800_{\pm 2.482}$	$7.800_{\pm 1.470}$	$7.800_{\pm 2.227}$	$8.200_{\pm 1.470}$	$6.800_{\pm 1.939}$	$6.800_{\pm 1.720}$	$\mathbf{3.000}_{\pm 0.632}$	$\underline{3.400}_{\pm 2.059}$	$4.200_{\pm 2.227}$
F1	$3.667_{\pm 3.145}$	$6.333_{\pm 2.285}$	$8.333_{\pm 1.491}$	$8.000_{\pm 1.000}$	$4.167_{\pm 2.794}$	$6.667_{\pm 3.091}$	$7.500_{\pm 1.607}$	$3.833_{\pm 1.462}$	$\mathbf{3.167}_{\pm 1.344}$	$\underline{3.500}_{\pm 1.979}$
AUC	$4.667_{\pm 2.749}$	$5.667_{\pm 2.055}$	$8.667_{\pm 1.106}$	$7.833_{\pm 1.067}$	$4.833_{\pm 2.794}$	$7.500_{\pm 2.872}$	$7.333_{\pm 1.374}$	$\underline{2.500}_{\pm 1.500}$	$\mathbf{2.333}_{\pm 0.943}$	$3.833_{\pm 1.675}$
L_2 dist. of corr.	$4.818_{\pm 2.918}$	$5.636_{\pm 1.872}$	$8.091_{\pm 1.781}$	$7.909_{\pm 1.564}$	$7.000_{\pm 3.191}$	$6.909_{\pm 2.429}$	$6.818_{\pm 1.402}$	$\underline{2.727}_{\pm 1.286}$	$\mathbf{2.273}_{\pm 0.862}$	$3.000_{\pm 1.595}$
Detection score	$3.909_{\pm 3.502}$	$6.182_{\pm 1.696}$	$8.818_{\pm 1.466}$	$7.273_{\pm 1.213}$	$5.000_{\pm 3.045}$	$8.091_{\pm 2.391}$	$6.000_{\pm 1.595}$	$3.909_{\pm 1.164}$	$\mathbf{2.455}_{\pm 1.827}$	$\underline{3.545}_{\pm 1.725}$
JSD	$7.182_{\pm 2.167}$	$\mathbf{1.273}_{\pm 0.617}$	$8.182_{\pm 1.641}$	$8.818_{\pm 1.029}$	$6.909_{\pm 2.314}$	$7.000_{\pm 1.651}$	$6.545_{\pm 1.305}$	$\underline{2.455}_{\pm 1.076}$	$3.091_{\pm 0.793}$	$3.727_{\pm 1.052}$
WD	$\underline{3.091}_{\pm 3.315}$	$5.636_{\pm 1.611}$	$7.545_{\pm 1.827}$	$8.000_{\pm 1.477}$	$6.455_{\pm 3.144}$	$8.364_{\pm 1.823}$	$5.727_{\pm 2.339}$	$4.182_{\pm 1.466}$	$3.182_{\pm 1.192}$	$\mathbf{3.000}_{\pm 1.954}$
DCR	$6.000_{\pm 2.558}$	$6.182_{\pm 2.328}$	$8.455_{\pm 1.725}$	$6.182_{\pm 3.186}$	$4.455_{\pm 3.726}$	$6.545_{\pm 2.426}$	$5.909_{\pm 1.676}$	$4.091_{\pm 2.678}$	$\underline{3.818}_{\pm 2.124}$	$\mathbf{3.545}_{\pm 1.924}$

random forest and a catboost model, on the data-specific prediction task (the corresponding hyperparameter settings are reported in Appendix I). We compare the model-averaged real test performance, $\text{Perf}(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$, to the performance when trained on the synthetic data, $\text{Perf}(\mathcal{D}_{\text{gen}}, \mathcal{D}_{\text{test}})$. We subsample $\mathcal{D}_{\text{train}}$ in case of more than 50 000 observations to upper-bound the computational load. The results are averaged over ten different model seeds (in addition to the five random seeds that impact the sampling process). For regression tasks, we consider the RMSE and for classification tasks, the macro-averaged F1 and AUC scores. We only report $|\text{Perf}(\mathcal{D}_{\text{gen}}, \mathcal{D}_{\text{test}}) - \text{Perf}(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})|$ in the main part of this paper. An absolute difference close to zero, that is, synthetic and real data induce the same performance, indicates that the generative model performs well.

Detection score. For each generative model, we report the accuracy of a catboost model that is trained to distinguish between real and generated (fake) samples (Borisov et al., 2023; Liu et al., 2023; Zhang et al., 2024). First, we subsample the real data subsets, $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$, to a maximum of 25 000 data samples to limit evaluation time. Then, we construct $\mathcal{D}^{\text{detect}}_{\text{train}}, \mathcal{D}^{\text{detect}}_{\text{valid}}$ and $\mathcal{D}^{\text{detect}}_{\text{test}}$ with equal proportions of real and fake samples. We tune each catboost model on $\mathcal{D}^{\text{detect}}_{\text{valid}}$ and report the accuracy of the best-fitting model on $\mathcal{D}^{\text{detect}}_{\text{test}}$ (see Appendix H for details). A (perfect) detection score of 0.5 indicates the model is unable distinguish fake from real samples.

Statistical similarity. We aim to assess the statistical similarity between real and generated data at both the feature and sample levels. We largely follow Zhao et al. (2021) and compare: (1) the Jensen-Shannon divergence (JSD; Lin, 1991) to quantify the difference in categorical distributions, (2) the Wasserstein distance (WD; Ramdas et al., 2017) to quantify the difference in continuous distributions, and (3) the L_2 distance between pair-wise correlation matrices. We use the Pearson correlation coefficient for two continuous features, the Theil uncertainty coefficient for two categorical features, and the correlation ratio for mixed types. Similar metrics for the evaluation of statistical similarity have been used by Zhang et al. (2024).

Distance to closest record. That is, the minimum Euclidean distance of a generated data point to any observation in $\mathcal{D}_{\text{train}}$ (Borisov et al., 2023; Zhao et al., 2021). We one-hot encode categorical features and standardize all features to zero mean and unit variance to ensure each feature contributes equally to the distance. We compute the average distance to closest record (DCR) as a robust estimate. For brevity, we report the absolute difference of the DCR of the synthetic data and the DCR of the real test set. A good DCR value, indicating both realistic and sufficiently private data, should be close to zero.

4.2 Results

Table 1 shows the average rank of each generative model across all datasets for the considered metrics. The ranks in terms of the F1 and AUC scores are averaged over the classification task datasets. Likewise, the RMSE rank averages include the regression task datasets. We assign the maximum possible rank when a model could not be trained on a given dataset or could not be evaluated in reasonable time. This includes TabDDPM, which outputs NaNs for `acsincome` and `diabetes` and CoDi, which we consider to be prohibitively expensive to train on `diabetes` (estimated 14.5 hours) and `lending` (estimated 60 hours). Similarly, SMOTE is very inefficient in sampling for large datasets (78 min for 1000 samples on `acsincome` and 182 min on `covertype`) and does not finish the evaluation within 12 hours. The dataset-specific results (including standard errors) and average metrics over all datasets are detailed in Appendix R. We provide visualizations of the

Table 2: Ablation study for five CDTD configurations with progressive addition of model components. We report the median performance metrics over `acsincome`, `adult`, `beijing` and `churn`.

Config.	**A**	**B**	**C**	**D**	**CDTD (per type)**
RMSE (abs. diff.; ↓)	0.041	0.042	0.043	0.037	0.033
F1 (abs. diff.; ↓)	0.012	0.013	0.012	0.016	0.015
AUC (abs. diff.; ↓)	0.004	0.005	0.005	0.004	0.004
L_2 distance of corr. (↓)	0.131	0.124	0.146	0.118	0.127
Detection score (↓)	0.577	0.583	0.590	0.561	0.560
JSD (↓)	0.011	0.011	0.011	0.012	0.013
WD (↓)	0.004	0.005	0.005	0.003	0.003
DCR (abs. diff. to test; ↓)	0.405	0.361	0.386	0.299	0.372

captured correlations in the synthetic sample compared to the real training set in Appendix Q and distribution plots for a qualitative comparison in Appendix P.

Sample quality. CDTD consistently outperforms the considered benchmark models in most sample quality metrics. Specifically, we see a major performance edge in terms of the detection score, the L_2 distance of the correlation matrices and the regression-based metrics. Only for the Jensen-Shannon divergence ARF, a tree-based method that is expected to model categorical features particularly well, outperforms CDTD. Interestingly, CDTD performs similar to TabDDPM on F1 scores, but outperforms it dramatically for regression tasks. TabDDPM appears to favor modeling categorical features accurately, thereby sacrificing continuous features, as visualized in Appendix Q. TabSyn, a latent-space diffusion model, performs worse than CDTD and often TabDDPM, which define diffusion in data space. In Appendix M, we further compare CDTD and TabSyn and investigate the benefits of defining a diffusion model in data space. By utilizing score interpolation, CDTD is able to model intricate correlation structure more accurately than other frameworks. Most importantly, type-specific noise schedules mostly outperform the feature-specific and single noise schedule variants. This illustrates the importance of accounting for the high heterogeneity in tabular data on the feature type level. The different noise schedules per feature, however, appear to force too many constraints on the model and thus, decrease sample quality. Per-feature noise schedules would require more training steps to converge, as can be seen in Appendix O.

Training and sampling time. Figure 3 shows the average wall-clock time over all (for all models feasible) datasets for training as well as the time for sampling 1000 data points for each baseline model and the per feature CDTD variant (see Appendix T for details). We exclude SMOTE due to its considerably longer sampling with an average of 1377 seconds for 1000 samples. CDTD's use of embeddings (instead of one-hot encoding) for categorical features drastically reduces training times and thus, improves scaling to increasing number of categories. The ODE formulation of the diffusion process implies competitive sampling speeds, in particular compared to the diffusion-based benchmarks CoDi, TabDDPM and TabSyn. Despite TabSyn utilizing a separately trained encoder, this does *not* result in a lower-dimensional latent space and therefore, does not speed up sampling.

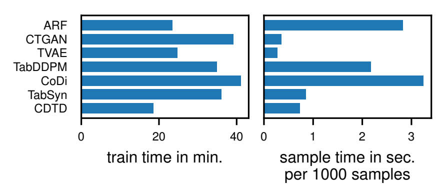

Figure 3: Average training and sampling wall-clock time for 1000 samples (excl. `acsincome`, `diabetes`, `lending`).

Ablation study. We conduct an ablation study over four datasets to investigate the separate components of our CDTD framework. The results are given in Table 2 (detailed results are in Appendix S). The baseline model *Config. A* includes a single noise schedule with the original piece-wise linear formulation (Dieleman et al., 2022) without loss normalization, improved model initialization or adaptive normalization, and the CE and MSE losses are naively averaged. Note that this configuration still is a novel contribution to the literature. *Config. B* adds our feature homogenization (i.e., loss normalization, improved initialization and adaptive normalization schemes), *Config. C* adds our proposed functional form for a single noise schedule with uniform initialization, and *Config. D* imposes per-type noise schedules. Lastly, we add the suggested (low noise level) overweighting timewarping initialization to arrive at the full CDTD (*per type*) model. We see the switch from the piece-wise linear functional form to our more robust noise schedule variant slightly harms sample quality. However, the per-type variant and the more informed initialization scheme compensate for this difference. Main differences are in the RMSE and detection score as well as training efficiency (the loss calibration and improved initialization facilitate model convergence). The final model especially works well on the larger datasets compared to the baseline (see Appendix S), as smaller datasets are relatively easy to fit with 3 million parameters, even without any model improvements. We investigate the sensitivity of CDTD to important hyperparameters in Appendix L.

5 Conclusion and Discussion

We propose a Continuous Diffusion model for mixed-type Tabular Data (CDTD) that combines score matching and score interpolation and imposes Gaussian diffusion processes on both continuous and embedded categorical features. We compared CDTD to various benchmark models and to a single noise schedule as typically used in image diffusion models. Our results indicate that addressing the high feature heterogeneity in tabular data on the feature type level and aligning type-specific diffusion elements, such as the noise schedules or losses, substantially benefits sample quality. Moreover, CDTD shows vastly improved scalability and can accommodate an arbitrary number of categories.

Our paper serves as an important step to customizing the diffusion probabilistic framework to tabular data. In particular, the common type of noise schedules allows for an easy to extend framework that might accelerate progress on diffusion models for tabular data. Crucially, CDTD allows the direct application of diffusion-related advances from the image domain, like classifier-free guidance, to tabular data without the need for a latent encoding. We leave further extensions to the tabular data domain, e.g., the exploration of accelerated sampling, efficient score model architectures, different forms of adaptive noise schedules, or the adaption to the data imputation task for future work.

Finally, we want to emphasize the potential misuse of synthetic data to support unwarranted claims. Any generated data should therefore not be blindly trusted, and synthetic data based inferences should always be compared to results from the real data. However, the correct use of generative models enables better privacy preservation and facilitates data sharing and open science practices.

Limitations

The main limitation of CDTD is the addition of hyperparameters, and tuning hyperparameters of a generative model can be a costly endeavor. However, our results also show that (1) a per type schedule is most often optimal and (2) our default hyperparameters perform well across a diverse set of datasets. Dieleman et al. (2022) show that the results of score interpolation for text data can be sensitive to the initialization of the embeddings. We have not encountered similar problems on tabular datasets (see Table 7). While the DCR indicates no privacy issues for the benchmark datasets used, additional caution must be taken when generating synthetic data from privacy sensitive sources. Lastly, for specific types of tabular data, such as time-series, our model may be outperformed by other generative models specialized for that type. While CDTD could be directly used for imputation using RePaint (Lugmayr et al., 2022), a separate training process is required to achieve the best results (Liu et al., 2024). Therefore, we leave the adaption of CDTD to the imputation task for future work.

Acknowledgements

This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-7437. We would also like to thank Sander Dieleman for helpful discussions.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. *arXiv preprint arXiv:2107.03006*, 2021.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language Models are Realistic Tabular Data Generators. In *International Conference on Learning Representations*, 2023.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models. In *Advances in Neural Information Processing Systems*, volume 35, New Orleans, USA, 2022.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. ISSN 1076-9757. doi: 10.1613/jair.953.

Song Chen. Beijing PM2.5 Data. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5JS49.

Ting Chen. On the Importance of Noise Scheduling for Diffusion Models. *arXiv preprint arXiv:2301.10972*, 2023.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

John Clore, Krzysztof Cios, Jon DeShazo, and Beata Strack. Diabetes 130-US hospitals for years 1999-2008. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5230J.

Lending Club. Loan data from Lending Club, 2015.

Partha Deb and Pravin K. Trivedi. Demand for Medical Care by the Elderly: A Finite Mixture Approach. *Journal of Applied Econometrics*, 12(3):313–336, 1997. ISSN 0883-7252, 1099-1255. doi: 10.1002/(SICI)1099-1255(199705)12:3<313::AID-JAE440>3.0.CO;2-G.

Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794. Curran Associates, Inc., 2021.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring Adult: New Datasets for Fair Machine Learning. *arXiv:2108.04884*, 2021.

Kelwin Fernandes, Pedro Vinagre, Paulo Cortez, and Pedro Sernadela. Online News Popularity. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5NS3V.

Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *arXiv preprint arXiv:2204.03458*, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12454–12465. Curran Associates, Inc., 2021.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887, Baltimore, Maryland, USA, 2022. PMLR.

Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

Allan Jabri, David Fleet, and Ting Chen. Scalable Adaptive Computation for Iterative Generation. *arXiv preprint arXiv:2212.11972*, 2022.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and Imputing Tabular Data via Diffusion and Flow-based Gradient-Boosted Trees. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238, Valencia, Spain, February 2024. PMLR.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. *arXiv preprint arXiv:2312.02696*, December 2023.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*. arXiv, April 2018.

A. Keramati, R. Jafari-Marandi, M. Aliannejadi, I. Ahmadian, M. Mozaffari, and U. Abbasi. Improved churn prediction in telecommunication industry using data mining techniques. *Applied Soft Computing*, 24:994–1012, 2014. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2014.08.041.

Jayoung Kim, Chaejeong Lee, and Noseong Park. STaSy: Score-based Tabular data Synthesis. *arXiv preprint arXiv:2210.04018*, 2023.

Diederik P. Kingma and Ruiqi Gao. Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation. *arXiv preprint arXiv:2303.00848*, September 2023.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. *arXiv preprint arXiv:2107.00630*, 2022.

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

Chaejeong Lee, Jayoung Kim, and Noseong Park. CoDi: Co-evolving Contrastive Diffusion Models for Mixed-type Tabular Synthesis. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, Honolulu, Hawaii, USA, 2023. PMLR.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM Improves Controllable Text Generation. *arXiv preprint arXiv:2205.14217*, 2022.

Jianhua Lin. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: Generative Modelling for Tabular Data by Learning Relational Structure. In *International Conference on Learning Representations*, 2023.

Yixin Liu, Ajanthan Thalaiyasingam, Hisham Husain, and Vu Nguyen. Self-supervision improves diffusion models for tabular data imputation. *arXiv preprint arXiv:2407.18013*, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. In *36th Conference on Neural Information Processing Systems*, 2022.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11451–11461, New Orleans, LA, USA, 2022. IEEE. ISBN 978-1-66546-946-3. doi: 10.1109/CVPR52688.2022.01117.

Chao Ma, Sebastian Tschiatschek, José Miguel Hernández-Lobato, Richard Turner, and Cheng Zhang. VAEM: A Deep Generative Model for Heterogeneous Mixed Type Data. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11237–11247. Curran Associates, Inc., 2020.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete Score Matching: Generalized Score Matching for Discrete Data. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34532–34545. Curran Associates, Inc., 2022.

S. Moro, P. Rita, and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5K306.

Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139. PMLR, 2021.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410, Oct 2016. doi: 10.1109/DSAA.2016.49.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: Facilitating innovative use cases of synthetic data in different data modalities. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3173–3188. Curran Associates, Inc., January 2023.

Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests. *Entropy*, 19(2), 2017.

Florence Regol and Mark Coates. Diffusing Gaussian Mixtures for Generating Categorical Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9570–9578, 2023. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v37i8.26145.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*, 2022.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, Lille, France, 2015. JMLR.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*, 2021.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned Embedding Diffusion for Text Generation. *arXiv preprint arXiv:2211.04236*, 2022.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based Continuous-time Discrete Diffusion Models. *arXiv preprint arXiv:2211.16750*, 2023.

Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. ISSN 0899-7667, 1530-888X. doi: 10.1162/NECO_a_00142.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206, Valencia, Spain, 2023. PMLR.

Tong Wu, Zhihao Fan, Xiao Liu, Yeyun Gong, Yelong Shen, Jian Jiao, Hai-Tao Zheng, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. AR-Diffusion: Auto-Regressive Diffusion Model for Text Generation. *arXiv preprint arXiv:2305.09515*, 2023.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling Tabular Data using Conditional GAN. In *Advances in Neural Information Processing Systems*, volume 12. Curran Associates, Inc., 2019.

I-Cheng Yeh. Default of credit card clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *International Conference on Learning Representations*, Vienna, Austria, 2024. arXiv.

Zilong Zhao, Aditya Kunar, Hiek Van der Scheer, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective Table Data Synthesizing. In *Proceedings of the 13th Asian Conference on Machine Learning*, volume 157, pp. 97–112. PMLR, 2021.

A Loss Calibration

A priori, we let the model be indifferent between features, that is, we scale the loss of each feature such that at the terminal timestep the same loss is attained. Here, the signal-to-noise ratio is sufficiently low to approximate a situation in which the model has no information about the data. Thus, we are looking for calibrated losses $\mathcal{L}^*_{\text{MSE}}(x^{(i)}_{\text{cont}}, 1)$ and $\mathcal{L}^*_{\text{CE}}(x^{(j)}_{\text{cat}}, 1)$ which at $t = 1$ achieve unit loss in expectation.

For a single scalar feature and a given timestep t, we can write the empirical denoising score matching loss (Equation (3)) in the EDM parameterization (Karras et al., 2022) as:

$$\mathcal{L}_{\text{MSE}}(x^{(i)}_{\text{cont}}, t) = \lambda(t)\Big(\underbrace{c_{\text{skip}}(t)x_t + c_{\text{out}}(t)F^{(i)}_{\boldsymbol{\theta}}}_{s_{\boldsymbol{\theta}}(x_t,t)} - x^{(i)}_{\text{cont}}\Big)^2,$$

where $F^{(i)}_{\boldsymbol{\theta}}$ denotes the neural network output for feature i that parameterizes the score model $s_{\boldsymbol{\theta}}$. The parameters $c_{\text{skip}}(t) = \sigma^2_{\text{data}}/(\sigma^2(t) + \sigma^2_{\text{data}})$ and $c_{\text{out}}(t) = \sigma(t) \cdot \sigma_{\text{data}}/(\sqrt{\sigma^2(t) + \sigma^2_{\text{data}}})$ depend on $\sigma(t)$ (and σ_{data}) and therefore on timestep t. For $t \to 1$, $\sigma(t)$ approaches the maximum noise level $\sigma_{\text{cont,max}}$ and $c_{\text{skip}}(t) \to 0$ and $c_{\text{out}}(t) \to 1$ such that the score model directly predicts the data at high noise levels. For $t \to 0$, the model shifts increasingly towards predicting the error that has been added to the true data. In the EDM parameterization, the explicit timestep weight (used to achieve a unit loss across timesteps at initialization, see Appendix B) is $\lambda(t) = 1/c_{\text{out}}(t)^2 \approx 1$ for $t = 1$.

At the terminal timestep $t = 1$, we now have:

$$\begin{aligned}
\mathbb{E}_{p(x^{(i)}_{\text{cont}})}[\mathcal{L}_{\text{MSE}}(x^{(i)}_{\text{cont}}, 1)] &= \lambda(1)\,\mathbb{E}_{p(x^{(i)}_{\text{cont}})}\Big(c_{\text{skip}}(1)x_1 + c_{\text{out}}(1)F^{(i)}_{\boldsymbol{\theta}} - x^{(i)}_{\text{cont}}\Big)^2, \\
&\approx \mathbb{E}_{p(x^{(i)}_{\text{cont}})}\Big(0 \cdot x_1 + 1 \cdot F^{(i)}_{\boldsymbol{\theta}} - x^{(i)}_{\text{cont}}\Big)^2, \\
&= \mathbb{E}_{p(x^{(i)}_{\text{cont}})}\Big(F^{(i)}_{\boldsymbol{\theta}} - x^{(i)}_{\text{cont}}\Big)^2.
\end{aligned}$$

Without information, it is optimal to always predict the average value $\mathbb{E}_{p(x^{(i)}_{\text{cont}})}[x^{(i)}_{\text{cont}}]$ and thus, the minimum expected loss becomes:

$$\mathbb{E}_{p(x^{(i)}_{\text{cont}})}[\mathcal{L}_{\text{MSE}}(x^{(i)}_{\text{cont}}, 1)] = \mathbb{E}_{p(x^{(i)}_{\text{cont}})}\Big(\mathbb{E}_{p(x^{(i)}_{\text{cont}})}[x^{(i)}_{\text{cont}}] - x^{(i)}_{\text{cont}}\Big)^2 = \text{Var}[x^{(i)}_{\text{cont}}]\,.$$

Therefore, we have $\mathcal{L}^*_{\text{MSE}}(x^{(i)}_{\text{cont}}, 1) = \mathcal{L}_{\text{MSE}}(x^{(i)}_{\text{cont}}, 1)$ as long as we standardize $x^{(i)}_{\text{cont}}$ to unit variance.

For a single categorical feature, $x^{(j)}_{\text{cat}}$ is distributed according to the proportions p_c (for categories $c = 1, \ldots, C$). The denoising model for score interpolation is trained with the CE loss:

$$\mathcal{L}_{\text{CE}}(x^{(j)}_{\text{cat}}, t) = -\sum_{c=1}^{C} I(x^{(j)}_{\text{cat}} = c) \log F^{(j)}_{\boldsymbol{\theta},c}\,,$$

where $F^{(j)}_{\boldsymbol{\theta},c}$ denotes the score model's prediction of the class probability at timestep t. Without information, it is optimal to assign the c-th category the same proportion as in the training set. At $t = 1$, we thus let $F^{(j)}_{\boldsymbol{\theta},c} = p_c$ such that the minimum loss equals:

$$\begin{aligned}
\mathbb{E}_{p(x^{(j)}_{\text{cat}})}[\mathcal{L}_{\text{CE}}(x^{(j)}_{\text{cat}}, 1)] &= -\mathbb{E}_{p(x^{(j)}_{\text{cat}})}\sum_{c=1}^{C} I(x^{(j)}_{\text{cat}} = c) \log F^{(j)}_{\boldsymbol{\theta},c}\,, && (9) \\
&= -\sum_{c=1}^{C} \mathbb{E}_{p(x^{(j)}_{\text{cat}})}[I(x^{(j)}_{\text{cat}} = c) \log p_c]\,, && (10) \\
&= -\sum_{c=1}^{C} p_c \log p_c. && (11)
\end{aligned}$$

We use the training set proportions to compute the normalization constant $Z_j = -\sum_{c=1}^{C} p_c \log p_c$ to calibrate the loss for categorical features. Then,

$$\mathbb{E}_{p(x^{(j)}_{\text{cat}})}[\mathcal{L}^*_{\text{CE}}(x^{(j)}_{\text{cat}}, 1)] = \mathbb{E}_{p(x^{(j)}_{\text{cat}})}[\mathcal{L}_{\text{CE}}(x^{(j)}_{\text{cat}}, 1)/Z_j] = 1\,.$$

We have thus achieved calibrated losses with respect to the terminal timestep $t = 1$, that is, $\mathbb{E}_{p(x_{\text{cont}}^{(i)})}[\mathcal{L}^*_{\text{MSE}}(x_{\text{cont}}^{(i)}, 1)] = \mathbb{E}_{p(x_{\text{cat}}^{(j)})}[\mathcal{L}^*_{\text{CE}}(x_{\text{cat}}^{(j)}, 1)] = 1$ for all continuous features i and categorical features j.

B Output Layer Initialization

At initialization, we want the neural network to reflect the state of no information (see Appendix A). Likewise, our goal is a loss of one across all features and timesteps.

For continuous features i, we initialize the output layer weights (and biases) to zero such that the output of the score model for a single continuous feature, $F_{\boldsymbol{\theta}}^{(i)}$, is also zero. Since we use the EDM parameterization (Karras et al., 2022), we apply the associated explicit timestep weight $\lambda(t) = \frac{\sigma^2(t)+\sigma^2_{\text{data}}}{(\sigma(t)\cdot\sigma_{\text{data}})^2}$. This is explicitly designed to achieve a unit loss across timesteps at initialization and we show this analytically below. We denote the variances of the data $x_{\text{cont}}^{(i)}$ and of the Gaussian noise ϵ at time t as σ^2_{data} and $\sigma^2(t)$, respectively.

$$
\begin{aligned}
\mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}[\mathcal{L}^*_{\text{MSE}}(x_{\text{cont}}^{(i)}, t)] &= \lambda(t)\, \mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(c_{\text{skip}}(t)(x_{\text{cont}}^{(i)}+\epsilon) + c_{\text{out}}(t)F_{\boldsymbol{\theta}}^{(i)} - x_{\text{cont}}^{(i)}\Big)^2, \\
&= \lambda(t)\, \mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(c_{\text{skip}}(t)(x_{\text{cont}}^{(i)}+\epsilon) - x_{\text{cont}}^{(i)}\Big)^2, \\
&= \frac{\sigma^2(t)+\sigma^2_{\text{data}}}{(\sigma(t)\cdot\sigma_{\text{data}})^2}\mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(\frac{\sigma^2_{\text{data}}}{\sigma^2(t)+\sigma^2_{\text{data}}}(x_{\text{cont}}^{(i)}+\epsilon) - x_{\text{cont}}^{(i)}\Big)^2, \\
&= \frac{\sigma^2(t)+\sigma^2_{\text{data}}}{(\sigma(t)\cdot\sigma_{\text{data}})^2}\mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(\frac{\sigma^2_{\text{data}}\epsilon - \sigma^2(t)x_{\text{cont}}^{(i)}}{\sigma^2(t)+\sigma^2_{\text{data}}}\Big)^2, \\
&= \frac{1}{\sigma^2(t)+\sigma^2_{\text{data}}}\mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(\frac{\sigma_{\text{data}}}{\sigma(t)}\epsilon - \frac{\sigma(t)}{\sigma_{\text{data}}}x_{\text{cont}}^{(i)}\Big)^2, \\
&= \frac{1}{\sigma^2(t)+\sigma^2_{\text{data}}}\mathbb{E}_{p(x_{\text{cont}}^{(i)}),p(\epsilon)}\Big(\frac{\sigma^2_{\text{data}}}{\sigma^2(t)}\epsilon^2 + \frac{\sigma^2(t)}{\sigma^2_{\text{data}}}(x_{\text{cont}}^{(i)})^2 - 2\epsilon x_{\text{cont}}^{(i)}\Big), \\
&= \frac{1}{\sigma^2(t)+\sigma^2_{\text{data}}}\Big(\frac{\sigma^2_{\text{data}}}{\sigma^2(t)}\underbrace{\text{Var}(\epsilon)}_{\sigma^2(t)} + \frac{\sigma^2(t)}{\sigma^2_{\text{data}}}\underbrace{\text{Var}(x_{\text{cont}}^{(i)})}_{\sigma^2_{\text{data}}} - 2\underbrace{\text{Cov}(\epsilon, x_{\text{cont}}^{(i)})}_{0}\Big), \\
&= \frac{1}{\sigma^2(t)+\sigma^2_{\text{data}}}\Big(\sigma^2_{\text{data}} + \sigma^2(t)\Big) = 1.
\end{aligned}
$$

For categorical features j, we initialize the output layer such that the model achieves the respective losses under no information. Using the loss normalization constant Z_j (see Appendix A) and dropping the expectation over $p(\epsilon)$, we have

$$
\mathbb{E}_{p(x_{\text{cat}}^{(j)})}[\mathcal{L}^*_{\text{CE}}(x_{\text{cat}}^{(j)}, t)] = \mathbb{E}_{p(x_{\text{cat}}^{(j)})}[\mathcal{L}_{\text{CE}}(x_{\text{cat}}^{(j)}, t)/Z_j] = \frac{1}{Z_j}\mathbb{E}_{p(x_{\text{cat}}^{(j)})}[\mathcal{L}_{\text{CE}}(x_{\text{cat}}^{(j)}, t)].
$$

Hence, for $E_{p(x_{\text{cat}}^{(j)})}[\mathcal{L}_{\text{CE}}(x_{\text{cat}}^{(j)}, t)] = Z_j$, we obtain an expected loss of one irrespective of t. The neural network outputs a vector of logits $F_{\boldsymbol{\theta}}^{(j)}$ that are transformed into probabilities with a softmax function for each categorical feature. We denote the c-th element of that vector $\text{softmax}(\cdot)_c$. Since Z_j is derived in Equation (11) by imposing probabilities equal to the training set proportions for that category, p_c, we have

$$
\log p_c = \log \text{softmax}(F_{\boldsymbol{\theta}}^{(j)})_c = \log \frac{\exp(F_{\boldsymbol{\theta},c}^{(j)})}{\sum_{k=1}^{C}\exp(F_{\boldsymbol{\theta},k}^{(j)})} = F_{\boldsymbol{\theta},c}^{(j)} - \log\sum_{k=1}^{C}\exp(F_{\boldsymbol{\theta},k}^{(j)}).
$$

We initialize the neural network such that $F_{\boldsymbol{\theta},c}^{(j)} = \log p_c$ for all c. This is achieved by initializing the output layer weights to zero and the output layer biases to the relevant training set log-proportions of the corresponding class. Hence, this initialization gives us

$$F^{(j)}_{\boldsymbol{\theta},c} - \log \sum_{k=1}^{C} \exp(F^{(j)}_{\boldsymbol{\theta},k}) = \log p_c - \log \sum_{k=1}^{C} p_k = \log p_c,$$

which in turn leads to an initial loss of Z_j for all t and therefore achieves a uniform, calibrated loss of one at initialization similar to the continuous feature case.

C Adaptive Normalization of the Average Diffusion Loss

Both the loss calibration (see Appendix A) and output layer initialization (see Appendix B) ensure that the losses across timesteps (and features) are equal *at* initialization. During training, the adaptive noise schedules allow the model to focus automatically on the noise levels that matter most, i.e., where the loss increase is steepest. However, the better the model becomes at a given timestep t, the lower the loss at the respective timestep, and the lower the gradient signal relative to the signal for timesteps $\tilde{t} > t$. We counteract this with adaptive normalization of the average diffusion loss (averaged over the features) across timesteps. Specifically, we want to weight the average diffusion loss at timestep t, $\mathcal{L}(t)$ given in Equation (6), such that the normalized loss is the same (equal to one) for all t. Similar methods have been used by Karras et al. (2023) and Kingma & Gao (2023), we follow the latter in the setup of the corresponding network.

We train a neural network alongside our diffusion model to predict $\mathcal{L}(t)$ based on t and use the MSE loss to learn this weighting. First, we compute $c_{\text{noise}}(t) = \log(t)/4$ following the EDM parameterization (Karras et al., 2022). Then, we embed c_{noise} in frequency space (1024-dimensional) using Fourier features. The result is passed through a single linear layer to output a scalar value, passed through an exponential function to ensure that the prediction $\hat{\mathcal{L}}(t) \geq 0$. We initialize the weights and biases to zero, to ensure that at model initialization we have a unit normalization.

D Derivation of the Functional Timewarping Form

Since higher noise levels, σ, imply a lower signal-to-noise ratio, and in turn a larger loss, ℓ, we know that the loss must be a monotonically increasing and S-shaped function of the noise level. Additionally, the function has to be easy to invert and differentiate. We incorporate this prior information in the functional timewarping form of $F : \sigma \mapsto \ell$. A convenient choice is the cdf of the logistic distribution:

$$F_{\log}(y) = \big[1 + \exp\big(-\nu(y - \mu^*)\big)\big]^{-1}, \tag{12}$$

where μ^* describes the location of the inflection point of the S-shaped function and $\nu \geq 1$ indicates the steepness of the curve.

We let $y = \text{logit}(\sigma) = \log(\sigma/(1-\sigma))$ to change the domain of $F_{\log}$ from $(-\infty, \infty)$ to $(0, 1)$. The latter covers all possible values of the noise level σ scaled to $[0, 1]$ with the pre-specified minimum and maximum noise levels $\sigma_{\min}$ and $\sigma_{\max}$. To define the parameter μ in the same space and ensure that $0 < \mu < 1$, we also let $\mu^* = \text{logit}(\mu)$. Accordingly, we derive the cdf of the *domain-adapted* Logistic distribution:

$$F_{\text{d.a.log}}(\sigma) = \left[1 + \left(\frac{\sigma}{1-\sigma}\frac{1-\mu}{\mu}\right)^{-\nu}\right]^{-1}. \tag{13}$$

Since ℓ is not bounded, we introduce a multiplicative scale parameter, $\gamma > 0$, such that for timewarping we predict the potentially feature-specific loss as $\hat{\ell} = F(\sigma) = \gamma F_{\text{d.a.log}}(\sigma)$. $F_{\text{d.a.log}}$ can also be initialized to the cdf of the uniform distribution with $\mu = 0.5$, $\nu \approx 1$ and $\gamma = 1$ such that all noise levels are initially equally weighted. However, an initial overweighting of lower noise levels is beneficial for tabular data (see also Section 3.4).

Likewise, we can derive the inverse cdf $F^{-1}_{\text{d.a.log}}(t)$, that is our mapping of interest from timestep t to noise level σ, in closed form:

$$\sigma = F^{-1}_{\text{d.a.log}}(t) = \text{sigmoid}(c), \text{ with } c = \ln\left(\frac{\mu}{1-\mu}\right) + \frac{1}{\nu}\ln\left(\frac{t}{1-t}\right). \tag{14}$$

When training the diffusion model, we learn the parameters of $F_{\text{d.a.log}}$ as well as γ by predicting the diffusion loss using $F(\sigma)$ and the noise levels scaled to $[0, 1]$. At the beginning of each training step, we then use the current state of the parameters and $F^{-1}_{\text{d.a.log}}$, with a sampled timestep $t \sim \mathcal{U}_{[0,1]}$ as input, to derive σ. To allow for *feature-specific*, adaptive noise schedules, we separately introduce $F_k(\sigma_k)$ for each feature k, to predict the feature-specific loss ℓ_k based on the feature-specific scaled noise level σ_k.

Note that with timewarping we create a feedback loop in which we generate more and more σs from the region of interest, decreasing the number of observations available to learn the parameters in different noise level regions. We thus weight the timewarping loss, $||\ell - \hat{\ell}||_2^2$, when fitting $F(\sigma)$ to the data by the reciprocal of the pdf $f_{\text{d.a.log}}(\sigma)$ to mitigate this adverse effect (see Dieleman et al., 2022). Again, this function is available to us in closed form. With F_{log} and f_{log} denoting the respective cdf and pdf of the Logistic distribution, we have

$$\begin{aligned} f_{\text{d.a.log}}(\sigma) &= \frac{\partial}{\partial y} F_{\text{log}}(y)\bigg|_{y=\text{logit}(\sigma)} \frac{\partial}{\partial \sigma} \ln \frac{\sigma}{1-\sigma} \\ &= f_{\text{log}}(\text{logit}(\sigma)) \frac{1}{\sigma(1-\sigma)} \\ &= \frac{\nu}{\sigma(1-\sigma)} \cdot \frac{Z(\sigma,\mu,\nu)}{\big(1+Z(\sigma,\mu,\nu)\big)^2}, \end{aligned}$$

where we defined $Z(\sigma, \mu, \nu) = \left(\frac{\sigma}{1-\sigma}\frac{1-\mu}{\mu}\right)^{-\nu}$ and used the definitions of f_{log} and the parameter μ^*.

E Benchmark Datasets

Our selected benchmark datasets are highly diverse, particularly in the number of categories for categorical features (see Table 3). For the `diabetes` and `covertype` datasets, we transform the original multi-class classification problem into a binary classification task for ease of presentation. For the `covertype` data, the task is converted into predicting whether a forest of type 2 is present in a given 30 $\times$ 30 meter area. In the `diabetes` data, we convert the task by predicting whether a patient was readmitted to a hospital. All datasets are publicly accessible and (except `nmes`) licensed under creative commons.

Table 3: Overview of the selected experimental datasets. We count the outcome towards the respective features that remain after removing continuous features with an excessive number of missings. The minimum and maximum number of categories are taken over all categorical features.

Dataset	License	Prediction task	Total no. observations	No. of features categorical	continuous	No. of categories min.	max.
`acsincome` (Ding et al., 2021)	CC0	regression	1 664 500	8	3	2	529
`adult` (Becker & Kohavi, 1996)	CC BY 4.0	binary classification	48 842	9	6	2	42
`bank` (Moro et al., 2012)	CC BY 4.0	binary classification	41 188	11	10	2	12
`beijing` (Chen, 2017)	CC BY 4.0	regression	41 757	1	10	4	4
`churn` (Keramati et al., 2014)	CC BY 4.0	binary classification.	3 150	5	9	2	5
`covertype` (Blackard, 1998)	CC BY 4.0	binary classification	581 012	44	10	2	2
`default` (Yeh, 2016)	CC BY 4.0	binary classification	30 000	10	14	2	11
`diabetes` (Clore et al., 2014)	CC BY 4.0	binary classification	101 766	28	9	2	716
`lending` (Club, 2015)	DbCL 1.0	regression	9 182	10	34	2	3151
`news` (Fernandes et al., 2015)	CC BY 4.0	regression	39 644	14	46	2	2
`nmes` (Deb & Trivedi, 1997)	unknown	regression	4 406	8	11	2	4

F Baseline Models

Below, we give a brief description of our selected generative baseline models (including code sources).

SMOTE (Chawla et al., 2002) – a technique (not a generative model) typically used to oversample minority classes based on interpolation between ground-truth observations. We use SMOTENC for mixed-type data from the scikit-learn package and mostly adapt the code from the TabDDPM repository (Kotelnikov et al., 2023). For sampling, we utilize 16 CPU cores.

ARF (Watson et al., 2023) – a recent generative approach that is based on a random forest for density estimation. The implementation is available at `https://github.com/bips-hb/arfpy` and licensed under the MIT license. We use package version 0.1.1. For training, we utilize 16 CPU cores.

CTGAN (Xu et al., 2019) – one of the most popular Generative-Adversarial-Network-based models for tabular data. The implementation is available as part of the Synthetic Data Vault (Patki et al., 2016) at `https://github.com/sdv-dev/CTGAN` and licensed under the Business Source License 1.1. We use package version 0.9.0.

TVAE (Xu et al., 2019) – a Variational-Autoencoder-based model for tabular data. Similar to CTGAN. The implementation is available as part of the Synthetic Data Vault (Patki et al., 2016) at `https://github.com/sdv-dev/CTGAN` and licensed under the Business Source License 1.1. We use package version 0.9.0. Note that since we only use TVAE (and CTGAN) as benchmark, and do not provide a synthetic data creation service, the license permits the free usage.

TabDDPM (Kotelnikov et al., 2023) – a diffusion-based generative model for tabular data that combines multinomial diffusion (Hoogeboom et al., 2021) and diffusion in continuous space. An implementation is available as part of the `synthcity` package (Qian et al., 2023) at `https://github.com/vanderschaarlab/synthcity/` and licensed under the Apache 2.0 license. We use package version 0.2.7 with slightly adjusted code to allow for the manual specification of categorical features.

CoDi (Lee et al., 2023) – a diffusion model trained with an additional contrastive loss, and which factorizes the joint distribution of mixed-type tabular data into a distribution for continuous data conditional on categorical features and a distribution for categorical data conditional on continuous features. Similarly, the authors utilize the multinomial diffusion framework (Hoogeboom et al., 2021) to model categorical data. An implementation is available at `https://github.com/ChaejeongLee/CoDi` under an unknown license.

TabSyn (Zhang et al., 2024) – a diffusion-based model that first learns a tranformer-based VAE to map mixed-type data to a continuous latent space. Then, the diffusion model is trained on that latent space. We use the official code available at `https://github.com/amazon-science/tabsyn` under the Apache 2.0 license.

G Implementation Details

Each of the selected benchmark models requires a rather different, more specialized neural network architecture. Imposing the same architecture across models is therefore not possible. The same inability holds for the comparison of CDTD to other diffusion-based models: Our model is the first to use a continuous noise distribution on both continuous and categorical features, and therefore the alignment of important design choices, like the noise schedule, across models is not possible. In particular, the forward process of the multinomial diffusion framework (Hoogeboom et al., 2021) used in TabDDPM and CoDi, which is based on Markov transition matrices, does not translate to our setting.

To ensure a fair comparison in terms of sampling steps, we set the steps for CDTD, TabDDPM, CoDi and TabSyn to $\max(200, \text{default})$. We therefore increase the default number of sampling steps for CoDi and TabSyn (from 50 steps) and TabDDPM (from 100 steps for classification datasets). For TabDDPM and regression datasets, we use the suggested default of 1000 sampling steps.

We adjust each architecture to a total of ± 3 million trainable parameters on the `adult` dataset to improve the comparability further (see Table 4) and use the same architectures for all considered datasets. Note that the total number of parameters may vary slightly across datasets due to different number of features and categories affecting the onehot encoding but is still comparable across models.

Table 4: Total number of trainable parameters per model on the `adult` dataset.

Model	Trainable parameters
CTGAN	3 000 397
TVAE	2 996 408
TabDDPM	3 003 924
CoDi	2 998 043
TabSyn	3 001 646
CDTD (per type schedule, TabDDPM architecture)	2 999 721

We also align the embedding/bottleneck dimensions for CTGAN, TVAE, TabDDPM, TabSyn and CDTD to 256. To align TabDDPM, TabSyn and CDTD further, we use the TabDDPM architecture for all models, with appropriate adjustments for different input types and dimensions. If applicable, all models are trained for 30k steps on a single RTX 4090 instance, using PyTorch version 2.2.2.

Below, we briefly discuss our model-specific hyperparameter choices.

SMOTE (Chawla et al., 2002): We use the default hyperparameters suggested for the SMOTENC scikit-learn implementation.

ARF (Watson et al., 2023): We use the authors's suggested default hyperparameters. In particular, we use 20 trees, $\delta = 0$ and a minimum node size of 5. We follow the official package implementation and set the maximum number of iterations to 10 (see `https://github.com/bips-hb/arfpy`).

CTGAN (Xu et al., 2019): We follow the popular implementation in the Synthetic Data Vault package (see `https://github.com/sdv-dev/CTGAN`). For this model to work, the batch size must be divisible by 10. Therefore, we adjust the batch size if necessary. We use a 256-dimensional embedding (instead of the default embedding dimension of 128) to better align the CTGAN architecture with TVAE, TabDDPM, TabSyn and CDTD.

TVAE (Xu et al., 2019): We again follow the implementation in the Synthetic Data Vault. We use a 256-dimensional embedding to better align the architecture with CTGAN, TabDDPM, TabSyn and CDTD.

TabDDPM (Kotelnikov et al., 2023): There are no general default hyperparameters provided. Hence, we mostly adapt the papers' tuned hyperparameters for the `adult` dataset (one of the few used datasets that includes both continuous and categorical features). However, we decrease the learning rate from 0.002 to 0.001, since most of the tuned models in the paper used learning rates around 0.001. For regression task datasets, we use 1000 sampling steps in accordance with the author's settings. For classification task datasets, we use 200 sampling steps (instead of the default 100 steps), to better align the model with CoDi and CDCD. Note also that for classification task datasets, TabDDPM models the conditional distribution $p(x|y)$, instead of the unconditional distribution $p(x)$ which is modeled for regression tasks. We adjust the dimension of the bottleneck to 256 (instead of the default 128) to also accommodate also larger datasets and align the model with CTGAN, TVAE,and CDTD.

CoDi (Lee et al., 2023): We use the default hyperparameters from the official code (see `https://github.com/ChaejeongLee/CoDi`).

TabSyn (Zhang et al., 2024): We use the default hyperparameters as suggested by the authors. The training steps that go towards training the VAE and the denoising network follow the proportions given in the official code (see `https://github.com/amazon-science/tabsyn`). To improve comparability to TabDDPM, CoDi and CDTD, we use the same neural network architecture as TabDDPM, which only differs slightly from the original architecture. We leave the VAE untouched.

CDTD (ours): To ensure comparability in particular to TabDDPM, CoDi and TabSyn, we use the same neural network architecture as TabDDPM. We only change the input layers to accommodate our embedding-based framework. In the input layer, we vectorize all embedded categorical features and concatenate them with the scalar valued continuous features. The adjusted output layer ensures that we predict a single value for each continuous features and set of class-specific probabilities for each categorical feature. Since our use of embeddings introduces additional parameters, we scale the hidden layers slightly down relative to the TabDDPM to ensure approximately 3 million trainable parameters (instead of 808 neurons per layer we use 806) on the `adult` dataset. More details on the CDTD implementation are given in Appendix J.

H Tuning of the Detection Model

We use a catboost model (Prokhorenkova et al., 2018) to test whether real and generated samples can be distinguished. We generate the same number of fake observations for each of the real train, validation and test sets. We cap the maximum size of the real data subsets to 25 000, and subsample them if necessary, to limit the computational load. Per set, we combine real and fake observations to $\mathcal{D}^{\text{detect}}_{\text{train}}$, $\mathcal{D}^{\text{detect}}_{\text{valid}}$, and $\mathcal{D}^{\text{detect}}_{\text{test}}$, respectively. The catboost model is trained on $\mathcal{D}^{\text{detect}}_{\text{train}}$ with the task of predicting whether an observation is real or fake. We tune the catboost model with optuna and for 50 trials to maximize the accuracy on $\mathcal{D}^{\text{detect}}_{\text{valid}}$. The catboost hyperparameter search space is given in Table 5. Afterwards, we repeat the sampling process and the creation of $\mathcal{D}^{\text{detect}}_{\text{train}}$, $\mathcal{D}^{\text{detect}}_{\text{valid}}$ and $\mathcal{D}^{\text{detect}}_{\text{test}}$ for five different seeds. Each time, the model is trained on $\mathcal{D}^{\text{detect}}_{\text{train}}$ with the previously tuned hyperparameters, and evaluated on $\mathcal{D}^{\text{detect}}_{\text{test}}$. The average test set accuracy over the five seeds yields the estimated detection score.

Table 5: Catboost hyperparameter space settings. The model is tuned for 50 trials.

Parameter	**Distribution**
no. iterations	= 1000
learning rate	Log Uniform [0.001, 1.0]
depth	Cat([3,4,5,6,7,8])
L2 regularization	Uniform [0.1, 10]
bagging temperature	Uniform [0, 1]
leaf estimation iters	Integer Uniform [1, 10]

I Machine Learning Efficiency Models

For the group of machine learning efficiency models, we use the scikit-learn and catboost package implementations including the default parameter settings, if not specified otherwise below:

Logistic or Ridge Regression: max. iterations = 1000

Random Forest: max. depth = 12, no. estimators = 100

Catboost: no. iterations = 2000, early stopping rounds = 50, overfitting detector pval = 0.001

J CDTD Implementation Details

To enable a fair comparison to the other methods, and to TabDDPM and TabSyn in particular, the CDTD score model utilizes the exact same architecture and optimizer as Kotelnikov et al. (2023), which was also adapted by TabSyn (Zhang et al., 2024). An overview of the score model is provided in Figure 4: First, the noisy data, i.e., the noisy scalars for continuous features and the noisy embeddings

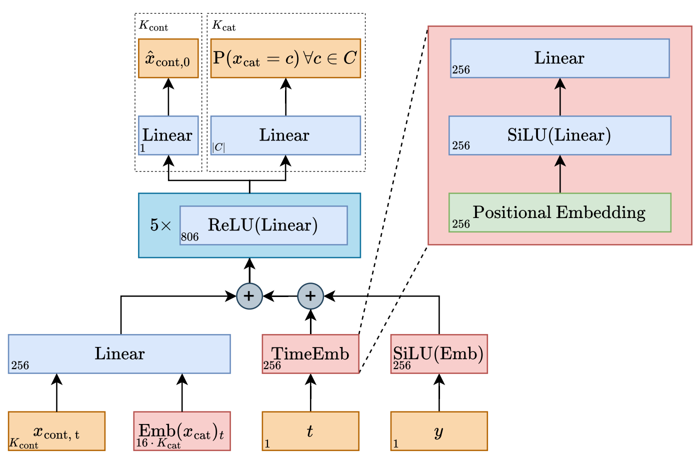

Figure 4: Overview of the CDTD architecture adapted from TabDDPM. The dimensions of the inputs and layer outputs are stated in the lower-left hand corner for a continuous features x_{cont} and a categorical features x_{cat}. Note that each categorical features can have a different number of categories $|C|$, impacting the output dimension of the final layer. Scalars are colored orange, embeddings red and linear layers blue. The positional embedding highlighted in green refers to the positional sinusoidal embedding. CDTD only conditions on y, i.e., the target feature, for classification task datasets.

for categorical features, are projected onto a 256-dimensional space. Similarly, timestep t and possibly conditioning information y are embedded in the same space. Then, all 256-dimensional vectors are added and the results is processed by a set of five fully-connected linear layers with ReLU activation functions. Lastly, a linear projection maps the output of the fully-connected layers to the required output dimensions, which depend on the number of features and number of categories per feature.

The only major difference to the TabDDPM setup are the inputs, as we need to embed the categorical features in Euclidean space. The output dimensions are the same, as we need to predict a single scalar for each $x_{\text{cont},i}$, and $|C_j|$ values for each $x_{\text{cat},j}$, with C_j the set of categories of feature j. We change the initialization of the output layer as described in Appendix B: To handle our inputs, we embed the categorical features in 16-dimensional space and add a feature-specific bias of the same dimension, which captures feature-specific information common to all categories and is initialized to zero. We L_2-normalize each embedding to prevent a degenerate embedding space in which embeddings are pushed further and further apart (see Dieleman et al., 2022). Also, Dieleman et al. (2022) argue that the standard deviation of the Normal distribution used to initialize the embeddings, denoted by σ_{init}, is an important hyperparameter. In this paper, we set $\sigma_{\text{init}} = 0.001$ for all datasets and have not seen detrimental effects. Table 7 indicates that CDTD is not sensitive to the choice of σ_{init}.

Since we utilize embeddings, we have to scale the neurons per layer slightly down in the stack of the five fully-connected layers (from 808 for TabDDPM to 806). Also, since TabDDPM samples discrete steps from $[0, T]$, with $T \gg 1$, we scale our timesteps $t \in [0, 1]$ up by 1000. We use the same optimizer (Adam), learning rate (0.001), learning rate decay (linear), EMA decay (0.999), and training steps (30000). However, since we work with embeddings we add a linear warmup schedule over the first 100 steps.

Instead of using the vanilla uniform (time)step sampling as the TabDDPM, the CDTD model uses antithetic sampling (Dieleman et al., 2022; Kingma et al., 2022). The timesteps are still uniformly distributed but spread out more evenly over the domain, which benefits the training of the adaptive noise schedules. For generation, we use an Euler sampler with 200 steps to minimize the discretization error.

K CDTD SAMPLING

To sample from our learned distribution, we need to run the reverse process of the probability flow ODE (Equation (2)). For example, for two different features x_1 and x_2, we deconstruct the ODE as:

$$
\begin{aligned}
\mathrm{d}\mathbf{x} &= -\frac{1}{2}\mathbf{G}(t)\mathbf{G}(t)^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\mathrm{d}t \\
&= -\begin{bmatrix} \dot{\sigma_1}(t)\sigma_1(t) & \\ & \dot{\sigma_2}(t)\sigma_2(t) \end{bmatrix} \begin{bmatrix} \frac{\hat{x_1} - x_1}{\sigma_1(t)^2} \\ \frac{\hat{x_2} - x_2}{\sigma_2(t)^2} \end{bmatrix} \mathrm{d}t \\
&= -\begin{bmatrix} \dot{\sigma_1}(t) & \\ & \dot{\sigma_2}(t) \end{bmatrix} \begin{bmatrix} \frac{\hat{x_1} - x_1}{\sigma_1(t)} \\ \frac{\hat{x_2} - x_2}{\sigma_2(t)} \end{bmatrix} \mathrm{d}t
\end{aligned}
$$

In practice, we use an Euler sampler with 200 discrete timesteps $\Delta t = t_{i+1} - t_i < 0$. The timesteps are generated as a linearly spaced grid on $[0, 1]$ and transformed afterwards into noise levels $\sigma_k(t)$ via the described timewarping procedure. For the discretized and simplified ODE above, this yields

$$
\mathbf{x}_{i+1} = \mathbf{x}_i - \begin{bmatrix} \frac{\Delta\sigma_1(t)}{\Delta t} & \\ & \frac{\Delta\sigma_2(t)}{\Delta t} \end{bmatrix} \begin{bmatrix} \frac{\hat{x_1} - x_1}{\sigma_1(t)} \\ \frac{\hat{x_2} - x_2}{\sigma_2(t)} \end{bmatrix} \Delta t = \mathbf{x}_i + \begin{bmatrix} \frac{x_1 - \hat{x_1}}{\sigma_1(t)} \\ \frac{x_2 - \hat{x_2}}{\sigma_2(t)} \end{bmatrix} \odot \begin{bmatrix} \Delta\sigma_1(t) \\ \Delta\sigma_2(t) \end{bmatrix}.
$$

where $\odot$ denotes the element-wise product. Hence, we are effectively taking *feature-specific* steps of length $\Delta\sigma_k(t)$. The adaptive noise schedules (timewarping) therefore not only affect the training process, but also focus most work in the reverse process on the noise levels that matter most for sample quality (i.e., where $\Delta\sigma_i(t)$ is small).

We use finite differences to approximate $\dot{\sigma}_i$, instead of the available, analytical variant, since $\frac{d\sigma_k(t)}{dt} \to \infty$ as $t \to 1$. The step Δt would therefore be required to decrease as $t \to 1$ to ensure $\Delta t \approx \mathrm{d}t$ holds. For a large number of steps, this assumption does not hold in practice, and for $\frac{\mathrm{d}\sigma_k(t)}{\mathrm{d}t}$ the update of $\mathbf{x}$ overshoots the target drastically. Intuitively, $\sigma_k(t)$ becomes too steep near the terminal timestep $t = 1$ such that the step size can not sufficiently compensate for the slope increase to turn $\frac{\mathrm{d}\sigma_k(t)}{\mathrm{d}t}$ into a good approximation of the actual change in $\sigma_k(t)$. Moreover, the analytical solution would approximate $\mathrm{d}\sigma_k(t) = \dot{\sigma}_k(t)\mathrm{d}t$, i.e., the change in the noise level caused by a change in t. Since we know *exactly* where $\sigma_k(t)$ will end up when changing t, we are better off using that exact value and let $\mathrm{d}\sigma_k(t) = \Delta\sigma_k(t)$. Table 6 shows that the gains in sample quality are marginal to non-existent after more than 500 sampling steps.

Table 6: Performance sensitivity of CDTD (*per type*) to increasing number of sampling steps. Each metric is averaged over five seeds. As a robust measure, we report the median over the ablation study datasets `acsincome`, `adult`, `beijing` and `churn` .

Steps	RMSE	F1	AUC	L_2 distance of corr.	Detection score	JSD	WD	DCR
200 (default)	0.033	0.015	0.004	0.127	0.560	0.013	0.003	0.372
500	0.028	0.018	0.005	0.130	0.565	0.012	0.003	0.372
1000	0.028	0.018	0.005	0.129	0.560	0.012	0.003	0.373
1500	0.028	0.018	0.005	0.129	0.561	0.012	0.003	0.374

L SENSITIVITY TO IMPORTANT HYPERPARAMETERS

The training and sampling processes of CDTD are affected by various novel hyperparameters. Generally, a per-type noise schedule works best as we show in our main results in Table 1 for a diverse set of benchmark datasets. Here, we examine the sensitivity of CDTD to two additional important hyperparameters: (1) the standard deviation of the noise used to initialize the embeddings (and therefore specific to score interpolation), σ_{init}, and (2) the weight of the low noise levels used to initialize the μ_k in the adaptive noise schedule parameterization.

The experiments in Dieleman et al. (2022) show that σ_{init} is a crucial hyperparameter for score interpolation on text data. The same sensitivity does not translate to the tabular data domain, as shown in our results in Table 7. The much smaller embedding dimension (16 vs. 256) and the *feature-specific* embeddings significantly decrease the number of distinguishable categories. Compared to a vocabulary size of 32000 for text data (Dieleman et al., 2022), we only face a maximum of 3151 categories in the `lending` dataset (see Table 3). Thus, unlike other generative (diffusion) models for tabular data, CDTD scales to a practically arbitrary number of categories.

Our proposed functional form for the adaptive noise schedules (see Appendix D) is the first to allow for the incorporation of prior information about the importance of low vs. high (normalized) noise levels. For this, we adjust the weight of low noise levels which directly determines the location of the inflection point μ_k (see Section 3.3). The results in Table 8 indicate low sample quality sensitivity to weight changes for a per-type noise schedule. The initialization only impacts the time to convergence but not (much) the location of the optimum. In our experiments, the number of training steps (30000) appears to be high enough for all model variants to converge.

Table 7: Performance sensitivity of CDTD (*per type*) to changes in the standard deviation σ_{init} in the initialization of the embeddings of categorical features. Each metric is averaged over five seeds. As a robust measure, we report the median over the ablation study datasets `acsincome`, `adult`, `beijing` and `churn` .

σ_{init}	RMSE	F1	AUC	L_2 distance of corr.	Detection score	JSD	WD	DCR
1	0.032	0.017	0.006	0.126	0.564	0.011	0.004	0.311
0.1	0.035	0.016	0.004	0.128	0.570	0.012	0.004	0.358
0.01	0.032	0.017	0.005	0.131	0.566	0.011	0.004	0.369
0.001 (default)	0.033	0.015	0.004	0.127	0.560	0.013	0.003	0.372

M ADVANTAGES OF DIFFUSION IN DATA SPACE

These days, inspired from diffusion models in the image and video domains, much work relies on the idea of latent diffusion. Here, we want to briefly discuss and emphasize that for tabular data, diffusion

Table 8: Performance sensitivity of CDTD (*per type*) to changes in the prior weight of low noise levels in the initialization of the adaptive noise schedules. Each metric is averaged over five seeds. As a robust measure, we report the median over the ablation study datasets `acsincome`, `adult`, `beijing` and `churn`.

Weight	RMSE	F1	AUC	L_2 distance of corr.	Detection score	JSD	WD	DCR
1	0.036	0.015	0.004	0.143	0.651	0.015	0.003	0.313
2	0.030	0.014	0.005	0.147	0.651	0.014	0.003	0.352
3 (default)	0.033	0.015	0.004	0.154	0.651	0.013	0.004	0.366
4	0.034	0.019	0.005	0.148	0.656	0.013	0.004	0.370

in latent space (represented by TabSyn) has important drawbacks and how CDTD, a diffusion model defined in data space differs from that.

Latent diffusion models first encode the data and map it into a latent space. The diffusion model itself is then trained in that latent space. Hence, the performance of the diffusion model directly depends on a second, separate model, with a separate training procedure. TabSyn uses a VAE model to encode mixed-type data into a common continuous space that is *not* lower-dimensional, so as to minimize reconstruction errors. Any reconstruction errors caused by the incapability of the VAE in turn reduce the sample quality of the eventually generated samples, no matter the capacity of the diffusion model. This suggests that we would want to train a high capable encoder/decoder, which adds additional training costs. Figure 3 shows that latent diffusion is not necessarily more efficient in the tabular data domain. In particular, if the latent space is not lower-dimensional to minimize reconstruction error, then sampling speed is not improved.

We further hypothesize that much tabular data, due to the lack of redundancy and spatial or sequential correlation, is difficult to summarize efficiently in a joint latent space. Hence, compared to other domains, larger VAEs and higher-dimensional latent spaces are required, increasing the training time. Also, there is the risk of the VAE not picking up on subtle correlations within the data or distorting existing correlations by mapping into the latent space. Any correlations not properly encoded in the latent space, cannot be learned or exploited by the diffusion model. Since we optimize the VAE on an *average* loss, its reconstruction and encoding performance of, for instance, minority classes or extreme values in long-tailed distributions is likely lacking. This makes the job of the diffusion model more difficult, if not impossible.

Lastly, we take great care in homogenizing categorical and continuous features throughout the training process (see Appendix A and B). This is a crucial part of modeling *mixed*-type data. Using a VAE to define a diffusion process in latent space only shifts the necessity for homogenization to the VAE training process. Not balancing different feature- or data-types and their losses induces an implicit importance weight for each feature. Thus, the VAE may sacrifice the reconstruction quality of some features in favor of others (Kendall et al., 2018; Ma et al., 2020).

To empirically investigate the difference of diffusion in data space (CDTD) and latent diffusion (TabSyn), we examine the worst *feature-specific* sample quality and other metrics that directly benefit from the model generating *all* features well. Our results in Table 9 show that, latent diffusion comes with a considerable decrease in sample quality (while imposing a similar architecture and number of parameters as well as sampling steps, see Appendix G). In particular, the attained maximum metrics indicate that TabSyn has issues modeling *all* features and their correlations sufficiently well. This supports our argument that a homogenization of data types is of crucial importance to avoid having the model implicitly favor one feature over another.

N Comparison to Related Work

Table 10 summarizes our comparison of CDTD to the selected diffusion-based benchmark models, that is, TabSyn, TabDDPM and CoDi. Of those models, only TabSyn applies diffusion in latent space, which comes with both advantages and costs (as discussed in Appendix M). TabSyn is the only other model besides CDTD that avoids one-hot encoding categorical features by using embeddings. This improves the scalability to a higher number of categories without blowing up the input dimensions. Although both models utilize embeddings, TabSyn's generative capabilities are more constrained by

Table 9: A comparison of the CDTD model to latent diffusion (TabSyn). We average each metric over five sampling seeds and as a robust measure report the median over the ablation datasets `acsincome`, `adult`, `beijing` and `churn`. Abs. diff. in corr. matrices refers to the absolute differences in the correlation matrices between ground truth and synthetic data. The maximum, minimum and mean are taken across features.

	Detection score	**L_2 dist. of corr.**	**JSD**			**WD**			**Abs. diff. in corr. matrices**	
			min	mean	max	min	mean	max	min	max
TabSyn	0.772	0.479	0.005	0.018	0.046	0.003	0.006	0.017	0	0.133
CDTD (per type)	0.566	0.131	0.001	0.012	0.022	0.001	0.003	0.007	0	0.052
improvement over TabSyn	1.364	3.656	5.000	1.500	2.091	3.000	2.000	2.429	0	2.558

jointly encoding all features in a latent space. As such, it is still less flexible than CDTD, in particular when modeling very unbalanced categorical data. Information on rare categories may easily be cut off in favor of attributing more capacity in the latent space to more prominent categories or features. It should also be noted that TabSyn is the only model that makes use of a Transformer architecture in its VAE, which means that it scales quadratically in the number of features and therefore may not be easily scaled to high-dimensional data.

The CDTD model is the first to utilize adaptive and type- or feature-specific noise schedules to model tabular data. Further, we take great care in homogenizing categorical and continuous features throughout the training process, including the model initialization (see Appendix A and B). No other model attempts balancing the different features types. This is problematic as it suggests that other models may suffer from feature-specific induced implicit importance weights that impact both training and generation processes. Hence, the sample quality of some features may be unintentionally sacrificed in favor of increasing the sample quality of other features (Kendall et al., 2018; Ma et al., 2020). Note that this also applies to TabSyn: Even though their diffusion model avoids this issue by relying on a single type of loss due to the continuous latent space, the VAE training process does not account for any balancing issues between the two data types. Hence, the balancing issue is not eliminated but got only shifted to the encoder VAE.

Lastly, CDTD and TabSyn are the only models that define the diffusion process in *continuous* space. As such, other advanced techniques, like classifier-free guidance or ODE/SDE samplers, can be directly applied. To accommodate categorical data, CoDi and TabDDPM make use of multinomial diffusion (Hoogeboom et al., 2021), which is an inherently *discrete* process and therefore prohibits such applications.

Table 10: Comparison of CDTD to the diffusion-based generative models CoDi, TabDDPM and TabSyn. ($*$) Note that the VAE trained as part of the TabSyn model does not balance type-specific losses, which induces an implicit weighting among features. This can worsen the sample quality of some features in favor of others.

	defined in feature space	**avoids one-hot encoding**	**balances feature types**	**adaptive noise schedule**	**type- or feature-specific noise schedules**	**diffusion in continuous space**
CoDi	✓					
TabDDPM	✓					
TabSyn		✓	$*$			✓
CDTD (ours)	✓	✓	✓	✓	✓	✓

O EXAMPLES OF LEARNED NOISE SCHEDULES

Next, we show the learned noise schedules for the smallest (`churn`) and the largest (`acsincome`) datasets. Additionally, we illustrate the fit of *single*, *per type* and *per feature* schedules to the respective losses.

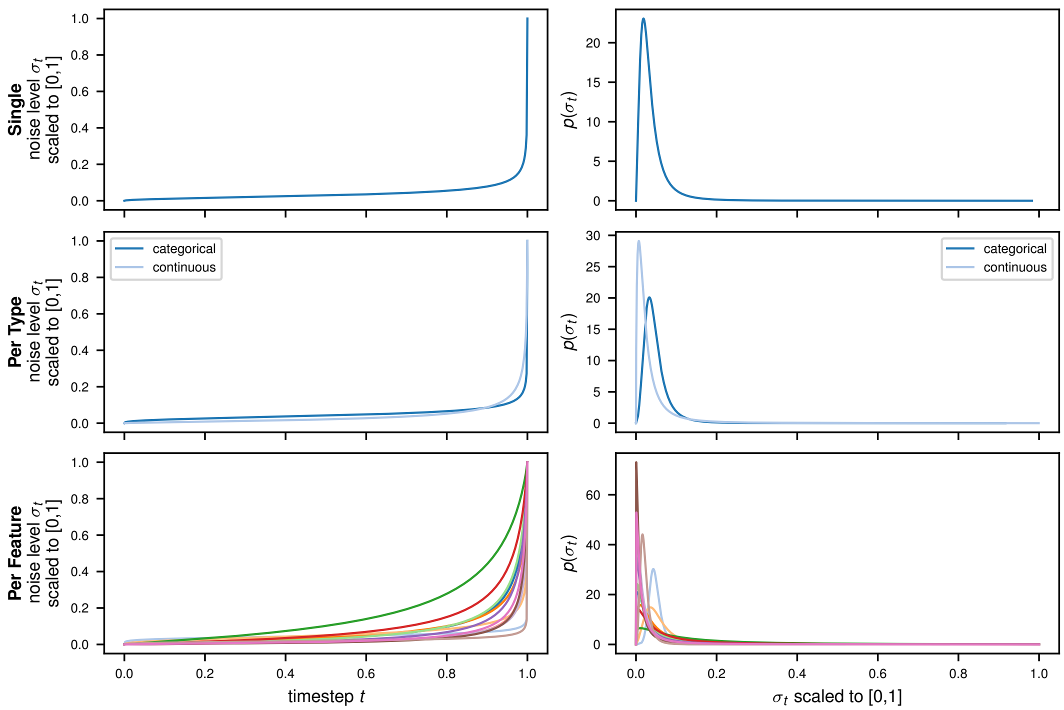

Figure 5: (Left): Learned noise schedules for `churn`. This reflects $F^{-1}_{\text{d.a.log},k}$. (Right): Implicit weighting of noise levels / timesteps. This visualizes $f_{\text{d.a.log},k}$.

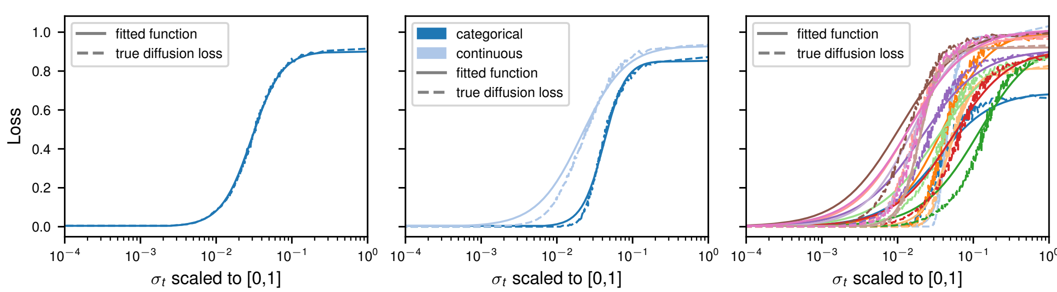

Figure 6: Illustration of the goodness of fit of the timewarping function F_k for single (left), per type (middle) and per feature noise schedules (right) on the `churn` data.

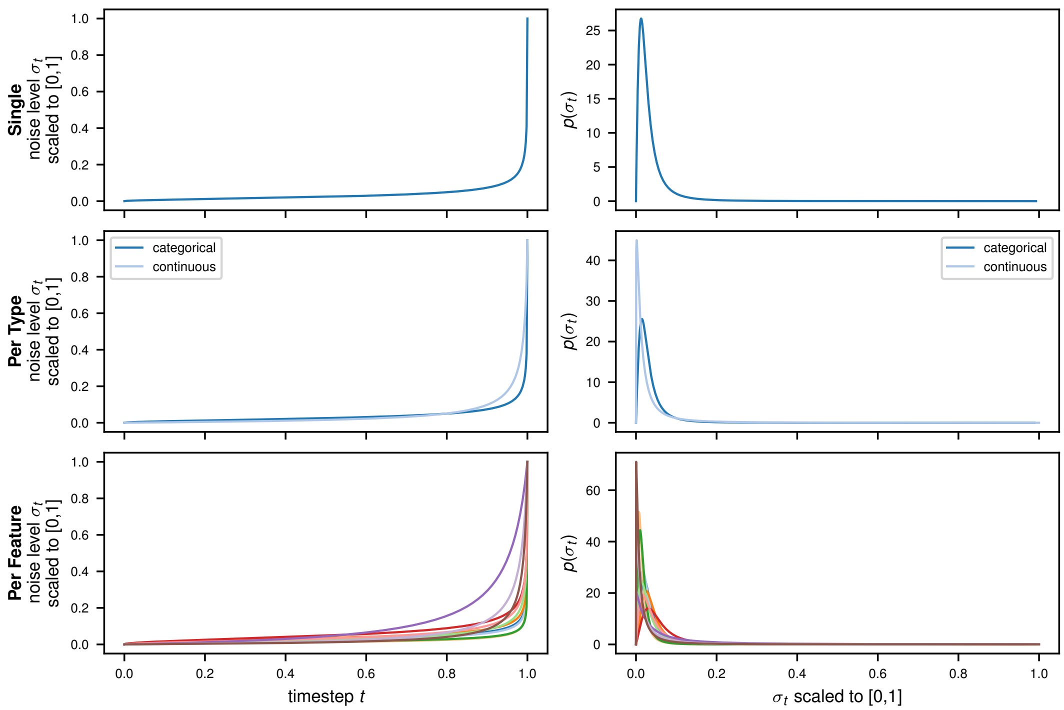

Figure 7: (Left): Learned noise schedules for `acsincome`. This reflects $F^{-1}_{\text{d.a.log},k}$. (Right): Implicit weighting of noise levels / timesteps. This visualizes $f_{\text{d.a.log},k}$.

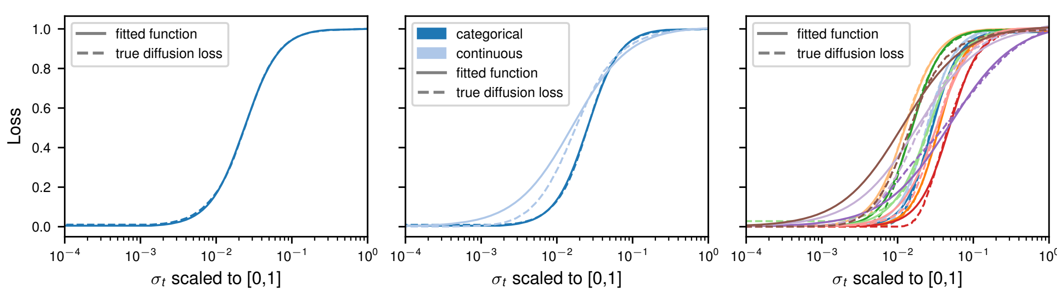

Figure 8: Illustration of the goodness of fit of the timewarping function F_k for single (left), per type (middle) and per feature noise schedules (right) on the `acsincome` data.

P QUALITATIVE COMPARISONS

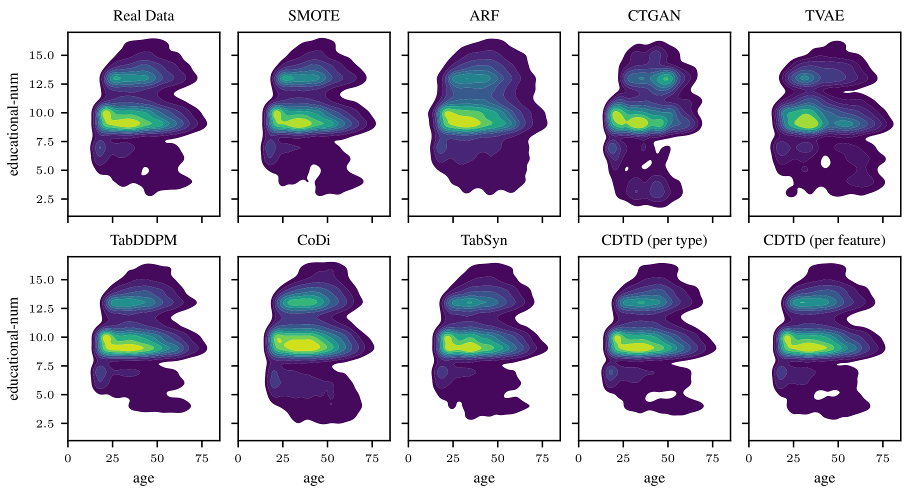

Figure 9: Bivariate density for age and educational-num from the `adult` data.

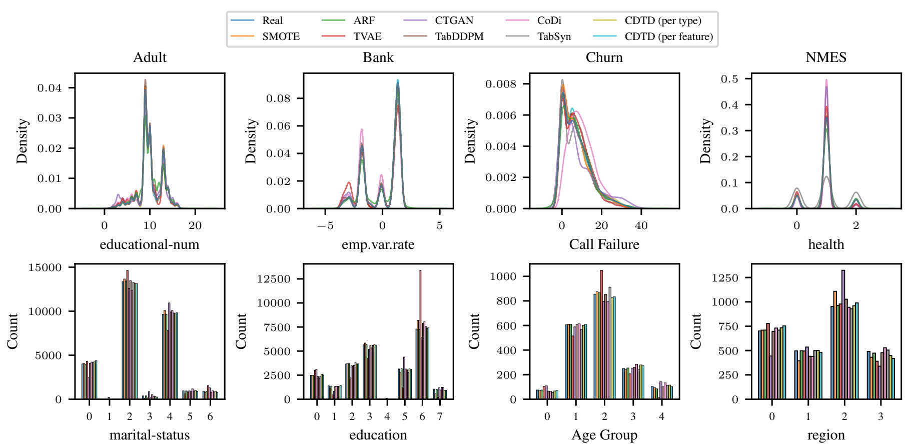

Figure 10: Comparison of some univariate distributions for `adult`, `bank`, `churn`, `nmes`.

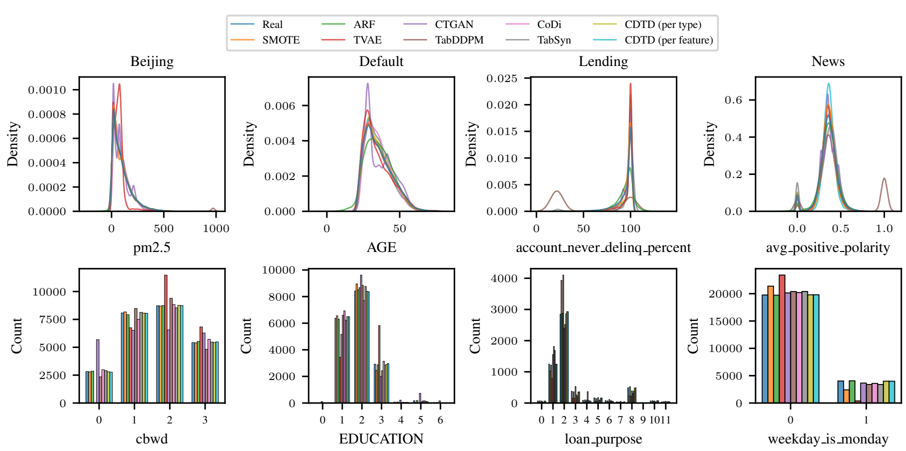

Figure 11: Comparison of some univariate distributions for `beijing`, `default`, `lending`, `news`. (Note that CoDi is prohibitively expensive to train on `lending` and therefore excluded.)

Q Visualizations of Captured Correlations

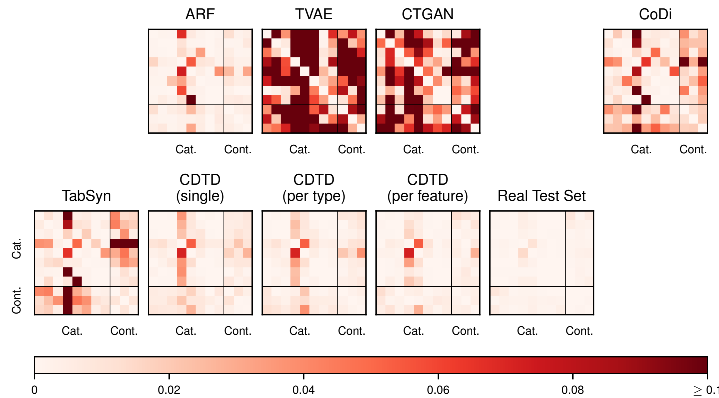

Figure 12: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `acsincome` dataset. TabDDPM generates NaNs for this dataset and is therefore excluded. SMOTE takes too long for sampling. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

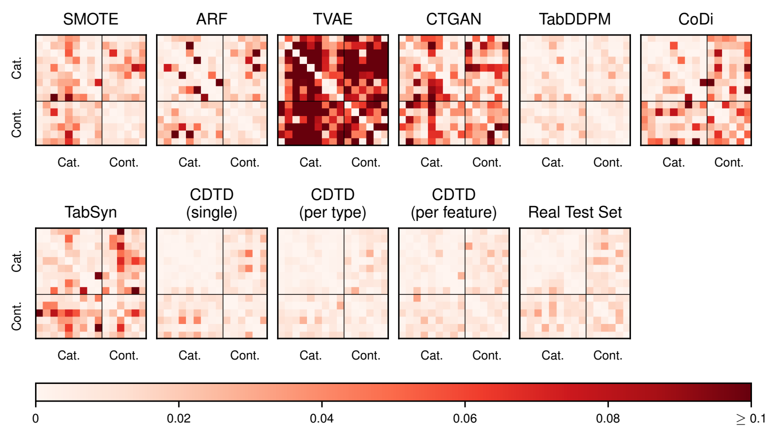

Figure 13: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `adult` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

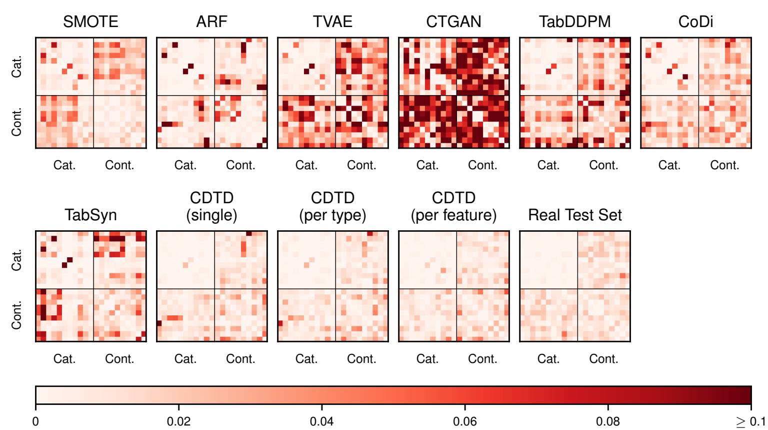

Figure 14: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `bank` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

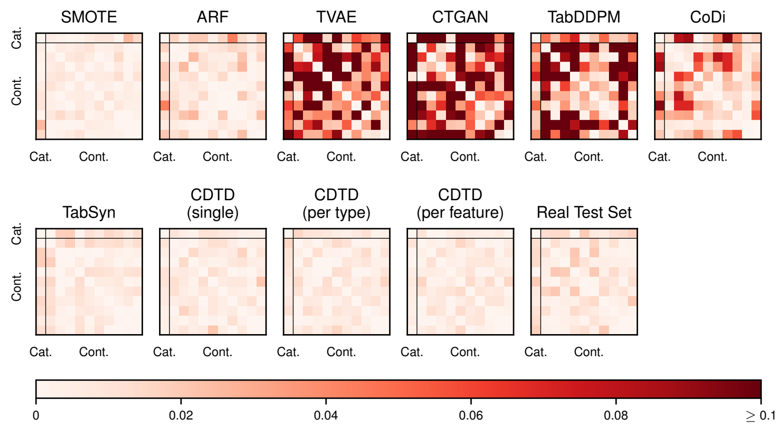

Figure 15: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `beijing` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

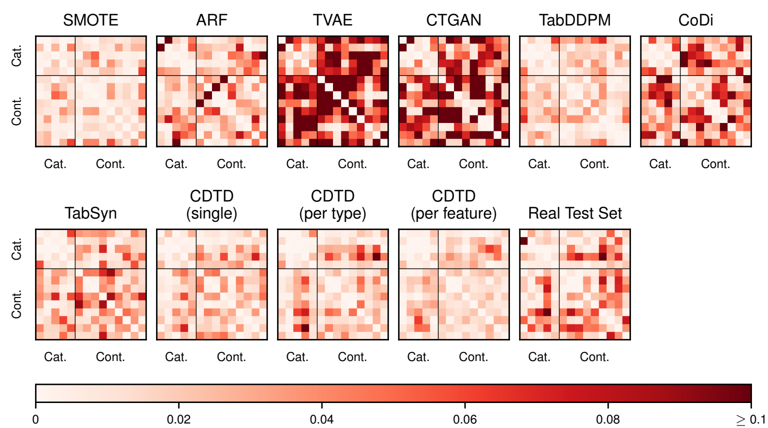

Figure 16: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `churn` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

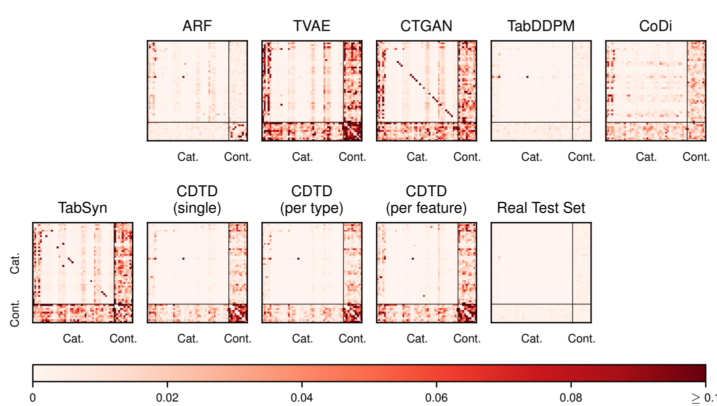

Figure 17: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `covertype` dataset. SMOTE takes too long for sampling. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

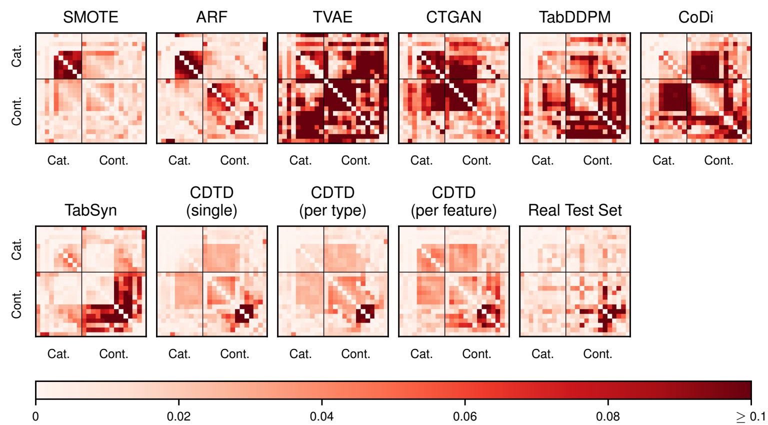

Figure 18: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `default` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

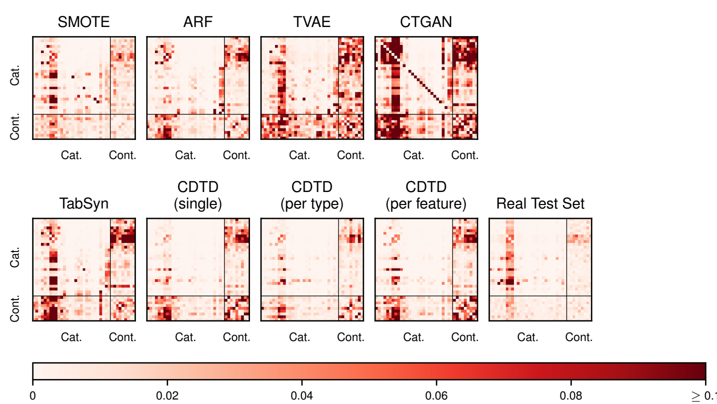

Figure 19: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `diabetes` dataset. TabDDPM generates NaNs for this dataset and is therefore excluded. CoDi is prohibitively expensive to train and therefore excluded. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

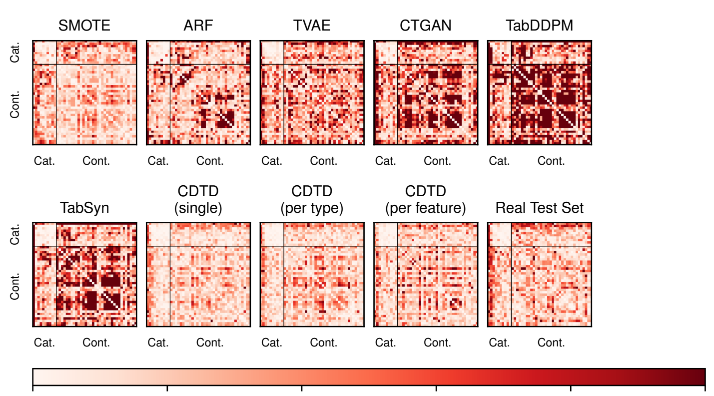

Figure 20: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `lending` dataset. CoDi is prohibitively expensive to train and therefore excluded. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

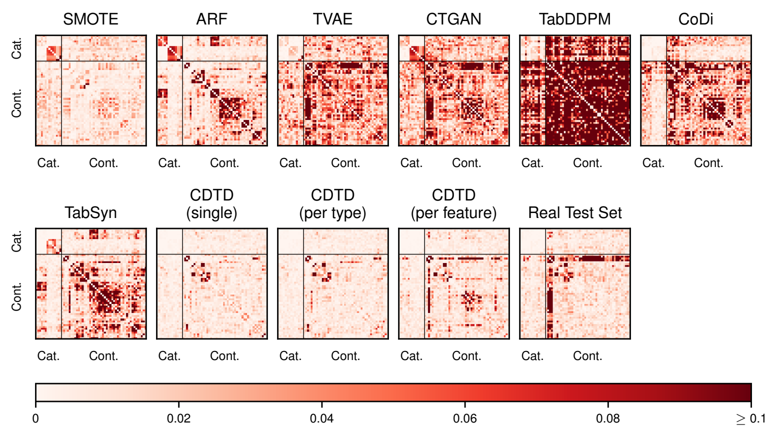

Figure 21: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `news` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

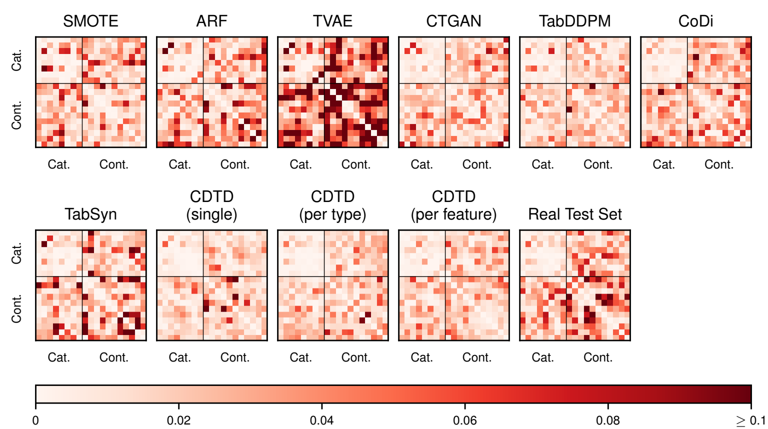

Figure 22: Element-wise absolute differences of the correlation matrices between the real training set and the synthetic data for the `nmes` dataset. Continuous (cont.) and categorical (cat.) features are indicated on the axes.

R DETAILED RESULTS

CoDi is prohibitively expensive to train on `lending` and `diabetes` and TabDDPM produces NaNs for `acsincome` and `diabetes`. SMOTE takes too long to sample datasets of a sufficient size for `acsincome` and `covertype` (see Table 29). For those models, the performance metrics on these datasets are therefore not reported. They are assigned a rank of 10 in Table 1 and are not taken into account when forming the average metrics reported in Table 11.

Table 11: Model evaluation results averaged over 11 datasets (skipping a dataset if the model was not trainable on it, which due to extensive sampling times for SMOTE includes two of the most complex datasets, `acsincome` and `covertype`) for seven benchmark models and for CDTD with three different noise schedules. Per performance metric, **bold** indicates the best, <u>underline</u> the second best result.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
RMSE (abs. diff.; ↓)	**0.083**	0.094	0.674	0.947	0.486	0.173	0.313	<u>0.084</u>	0.101	0.110
F1 (abs. diff.; ↓)	**0.007**	0.053	0.130	0.074	<u>0.015</u>	0.044	0.099	0.025	0.020	0.025
AUC (abs. diff.; ↓)	**0.008**	0.020	0.080	0.065	<u>0.009</u>	0.027	0.059	0.018	0.016	0.022
L_2 distance of corr. (↓)	0.866	1.321	2.187	2.745	3.786	1.200	2.025	<u>0.782</u>	**0.756**	0.990
Detection score (↓)	**0.661**	0.934	0.986	0.976	0.769	0.936	0.877	0.796	<u>0.768</u>	0.783
JSD (↓)	0.055	**0.011**	0.114	0.152	0.051	0.038	0.048	<u>0.015</u>	0.016	0.018
WD (↓)	**0.004**	0.011	0.023	0.025	0.061	0.022	0.016	0.010	<u>0.007</u>	0.009
DCR (abs. diff. to test; ↓)	1.278	1.588	3.336	1.621	**0.568**	1.000	2.593	0.796	0.806	<u>0.758</u>

Table 12: L_2 norm (incl. standard errors in subscripts) of the correlation matrix differences of real and synthetic train sets for seven benchmark models and for CDTD with three different noise schedules.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
`acsincome`	-	$0.242_{\pm 0.002}$	$1.696_{\pm 0.008}$	$1.136_{\pm 0.004}$	-	$0.517_{\pm 0.006}$	$0.524_{\pm 0.010}$	$0.141_{\pm 0.003}$	$0.129_{\pm 0.003}$	$0.119_{\pm 0.002}$
`adult`	$0.414_{\pm 0.016}$	$0.576_{\pm 0.006}$	$1.858_{\pm 0.010}$	$0.735_{\pm 0.012}$	$0.156_{\pm 0.006}$	$0.493_{\pm 0.009}$	$0.449_{\pm 0.011}$	$0.170_{\pm 0.007}$	$0.125_{\pm 0.009}$	$0.128_{\pm 0.010}$
`bank`	$0.404_{\pm 0.015}$	$0.819_{\pm 0.024}$	$0.947_{\pm 0.019}$	$2.758_{\pm 0.049}$	$0.898_{\pm 0.025}$	$0.499_{\pm 0.021}$	$0.677_{\pm 0.015}$	$0.323_{\pm 0.008}$	$0.266_{\pm 0.011}$	$0.256_{\pm 0.015}$
`beijing`	$0.081_{\pm 0.007}$	$0.133_{\pm 0.006}$	$1.445_{\pm 0.009}$	$1.642_{\pm 0.015}$	$1.133_{\pm 0.035}$	$0.363_{\pm 0.015}$	$0.096_{\pm 0.008}$	$0.075_{\pm 0.008}$	$0.073_{\pm 0.009}$	$0.071_{\pm 0.004}$
`churn`	$0.264_{\pm 0.036}$	$0.635_{\pm 0.026}$	$1.355_{\pm 0.043}$	$1.301_{\pm 0.041}$	$0.327_{\pm 0.044}$	$0.746_{\pm 0.062}$	$0.509_{\pm 0.053}$	$0.302_{\pm 0.041}$	$0.289_{\pm 0.043}$	$0.282_{\pm 0.044}$
`covertype`	-	$1.192_{\pm 0.017}$	$3.685_{\pm 0.005}$	$4.668_{\pm 0.003}$	$1.044_{\pm 0.001}$	$1.029_{\pm 0.032}$	$3.958_{\pm 0.243}$	$2.359_{\pm 0.011}$	$2.275_{\pm 0.009}$	$2.710_{\pm 0.009}$
`default`	$0.709_{\pm 0.048}$	$1.228_{\pm 0.021}$	$2.697_{\pm 0.021}$	$1.564_{\pm 0.029}$	$3.408_{\pm 0.105}$	$1.672_{\pm 0.061}$	$1.121_{\pm 0.042}$	$0.627_{\pm 0.068}$	$0.652_{\pm 0.102}$	$0.737_{\pm 0.033}$
`diabetes`	$2.355_{\pm 0.026}$	$1.189_{\pm 0.004}$	$1.654_{\pm 0.008}$	$5.351_{\pm 0.095}$	-	-	$2.381_{\pm 0.026}$	$1.201_{\pm 0.020}$	$0.803_{\pm 0.032}$	$1.345_{\pm 0.016}$
`lending`	$1.321_{\pm 0.063}$	$3.473_{\pm 0.057}$	$2.420_{\pm 0.016}$	$5.895_{\pm 0.026}$	$10.675_{\pm 0.015}$	-	$6.701_{\pm 0.034}$	$1.042_{\pm 0.075}$	$1.189_{\pm 0.040}$	$1.363_{\pm 0.097}$
`news`	$1.684_{\pm 1.466}$	$4.333_{\pm 0.128}$	$4.641_{\pm 0.028}$	$4.612_{\pm 0.016}$	$15.985_{\pm 0.081}$	$4.874_{\pm 0.148}$	$4.990_{\pm 0.024}$	$1.925_{\pm 0.527}$	$2.035_{\pm 0.475}$	$3.395_{\pm 0.950}$
`nmes`	$0.565_{\pm 0.047}$	$0.717_{\pm 0.054}$	$1.663_{\pm 0.035}$	$0.532_{\pm 0.030}$	$0.447_{\pm 0.031}$	$0.609_{\pm 0.032}$	$0.867_{\pm 0.046}$	$0.433_{\pm 0.025}$	$0.478_{\pm 0.083}$	$0.481_{\pm 0.058}$

Table 13: Jensen-Shannon divergence (incl. standard errors in subscripts) for seven benchmark models and for CDTD with three different noise schedules.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
`acsincome`	-	$0.013_{\pm 0.001}$	$0.256_{\pm 0.000}$	$0.309_{\pm 0.000}$	-	$0.076_{\pm 0.001}$	$0.045_{\pm 0.001}$	$0.025_{\pm 0.001}$	$0.024_{\pm 0.000}$	$0.022_{\pm 0.001}$
`adult`	$0.064_{\pm 0.001}$	$0.007_{\pm 0.001}$	$0.112_{\pm 0.001}$	$0.113_{\pm 0.001}$	$0.034_{\pm 0.001}$	$0.045_{\pm 0.001}$	$0.020_{\pm 0.001}$	$0.010_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.016_{\pm 0.000}$
`bank`	$0.039_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.086_{\pm 0.001}$	$0.191_{\pm 0.001}$	$0.029_{\pm 0.001}$	$0.038_{\pm 0.001}$	$0.054_{\pm 0.001}$	$0.010_{\pm 0.000}$	$0.009_{\pm 0.001}$	$0.012_{\pm 0.001}$
`beijing`	$0.006_{\pm 0.002}$	$0.005_{\pm 0.002}$	$0.147_{\pm 0.003}$	$0.257_{\pm 0.001}$	$0.035_{\pm 0.003}$	$0.018_{\pm 0.004}$	$0.007_{\pm 0.002}$	$0.003_{\pm 0.001}$	$0.005_{\pm 0.002}$	$0.005_{\pm 0.001}$
`churn`	$0.012_{\pm 0.004}$	$0.011_{\pm 0.004}$	$0.095_{\pm 0.003}$	$0.048_{\pm 0.004}$	$0.014_{\pm 0.004}$	$0.043_{\pm 0.001}$	$0.017_{\pm 0.002}$	$0.012_{\pm 0.003}$	$0.012_{\pm 0.002}$	$0.011_{\pm 0.002}$
`covertype`	-	$0.002_{\pm 0.000}$	$0.044_{\pm 0.000}$	$0.043_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.049_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.011_{\pm 0.000}$
`default`	$0.042_{\pm 0.001}$	$0.008_{\pm 0.001}$	$0.194_{\pm 0.001}$	$0.177_{\pm 0.001}$	$0.027_{\pm 0.002}$	$0.073_{\pm 0.002}$	$0.082_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.015_{\pm 0.001}$	$0.015_{\pm 0.001}$
`diabetes`	$0.067_{\pm 0.000}$	$0.009_{\pm 0.000}$	$0.093_{\pm 0.000}$	$0.187_{\pm 0.000}$	-	-	$0.095_{\pm 0.000}$	$0.022_{\pm 0.000}$	$0.023_{\pm 0.000}$	$0.026_{\pm 0.000}$
`lending`	$0.143_{\pm 0.001}$	$0.049_{\pm 0.002}$	$0.092_{\pm 0.001}$	$0.188_{\pm 0.001}$	$0.243_{\pm 0.002}$	-	$0.114_{\pm 0.002}$	$0.055_{\pm 0.001}$	$0.056_{\pm 0.001}$	$0.064_{\pm 0.002}$
`news`	$0.063_{\pm 0.001}$	$0.002_{\pm 0.001}$	$0.022_{\pm 0.001}$	$0.128_{\pm 0.001}$	$0.046_{\pm 0.000}$	$0.012_{\pm 0.001}$	$0.016_{\pm 0.001}$	$0.003_{\pm 0.001}$	$0.003_{\pm 0.001}$	$0.003_{\pm 0.001}$
`nmes`	$0.060_{\pm 0.001}$	$0.008_{\pm 0.002}$	$0.117_{\pm 0.002}$	$0.029_{\pm 0.003}$	$0.028_{\pm 0.004}$	$0.027_{\pm 0.003}$	$0.026_{\pm 0.001}$	$0.008_{\pm 0.001}$	$0.009_{\pm 0.001}$	$0.013_{\pm 0.003}$

Table 14: Wasserstein distance (incl. standard errors in subscripts) for seven benchmark models and for CDTD with three different noise schedules.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
`acsincome`	-	$0.007_{\pm 0.000}$	$0.037_{\pm 0.000}$	$0.021_{\pm 0.000}$	-	$0.017_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.001_{\pm 0.000}$	$0.001_{\pm 0.000}$
`adult`	$0.003_{\pm 0.000}$	$0.012_{\pm 0.000}$	$0.016_{\pm 0.000}$	$0.021_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.013_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$
`bank`	$0.002_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.021_{\pm 0.000}$	$0.040_{\pm 0.001}$	$0.011_{\pm 0.000}$	$0.030_{\pm 0.001}$	$0.005_{\pm 0.000}$	$0.006_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.004_{\pm 0.000}$
`beijing`	$0.002_{\pm 0.000}$	$0.009_{\pm 0.000}$	$0.021_{\pm 0.000}$	$0.058_{\pm 0.001}$	$0.011_{\pm 0.000}$	$0.019_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.002_{\pm 0.000}$
`churn`	$0.006_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.027_{\pm 0.001}$	$0.032_{\pm 0.001}$	$0.008_{\pm 0.002}$	$0.048_{\pm 0.002}$	$0.013_{\pm 0.002}$	$0.008_{\pm 0.001}$	$0.007_{\pm 0.001}$	$0.006_{\pm 0.001}$
`covertype`	-	$0.006_{\pm 0.000}$	$0.041_{\pm 0.000}$	$0.022_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.012_{\pm 0.000}$	$0.017_{\pm 0.000}$	$0.017_{\pm 0.000}$	$0.015_{\pm 0.000}$	$0.012_{\pm 0.000}$
`default`	$0.002_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.011_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.013_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$
`diabetes`	$0.004_{\pm 0.000}$	$0.012_{\pm 0.000}$	$0.020_{\pm 0.000}$	$0.038_{\pm 0.000}$	-	-	$0.011_{\pm 0.000}$	$0.038_{\pm 0.000}$	$0.020_{\pm 0.000}$	$0.042_{\pm 0.000}$
`lending`	$0.006_{\pm 0.000}$	$0.013_{\pm 0.001}$	$0.011_{\pm 0.000}$	$0.016_{\pm 0.000}$	$0.425_{\pm 0.001}$	-	$0.050_{\pm 0.000}$	$0.009_{\pm 0.000}$	$0.010_{\pm 0.000}$	$0.011_{\pm 0.000}$
`news`	$0.007_{\pm 0.000}$	$0.024_{\pm 0.000}$	$0.009_{\pm 0.000}$	$0.018_{\pm 0.000}$	$0.078_{\pm 0.001}$	$0.030_{\pm 0.000}$	$0.025_{\pm 0.000}$	$0.007_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.008_{\pm 0.000}$
`nmes`	$0.005_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.036_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.007_{\pm 0.001}$	$0.016_{\pm 0.001}$	$0.038_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.006_{\pm 0.000}$

Table 15: Detection score (incl. standard errors in subscripts) for seven benchmark models and for CDTD with three different noise schedules.

	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
acsincome	-	$0.808_{\pm0.001}$	$0.989_{\pm0.001}$	$0.985_{\pm0.000}$	-	$0.825_{\pm0.002}$	$0.680_{\pm0.002}$	$0.540_{\pm0.003}$	$0.532_{\pm0.004}$	$0.526_{\pm0.002}$
adult	$0.320_{\pm0.006}$	$0.889_{\pm0.002}$	$0.997_{\pm0.000}$	$0.967_{\pm0.001}$	$0.590_{\pm0.003}$	$0.992_{\pm0.001}$	$0.630_{\pm0.003}$	$0.604_{\pm0.002}$	$0.588_{\pm0.002}$	$0.591_{\pm0.005}$
bank	$0.633_{\pm0.008}$	$0.955_{\pm0.002}$	$1.000_{\pm0.000}$	$0.988_{\pm0.001}$	$0.783_{\pm0.003}$	$1.000_{\pm0.000}$	$0.843_{\pm0.002}$	$0.795_{\pm0.003}$	$0.739_{\pm0.003}$	$0.694_{\pm0.006}$
beijing	$0.976_{\pm0.001}$	$0.995_{\pm0.000}$	$0.998_{\pm0.000}$	$0.993_{\pm0.001}$	$0.966_{\pm0.002}$	$0.997_{\pm0.001}$	$0.966_{\pm0.001}$	$0.951_{\pm0.002}$	$0.949_{\pm0.001}$	$0.947_{\pm0.002}$
churn	$0.339_{\pm0.020}$	$0.853_{\pm0.002}$	$0.945_{\pm0.006}$	$0.843_{\pm0.011}$	$0.561_{\pm0.005}$	$0.730_{\pm0.012}$	$0.865_{\pm0.012}$	$0.621_{\pm0.016}$	$0.533_{\pm0.007}$	$0.544_{\pm0.031}$
covertype	-	$0.945_{\pm0.002}$	$0.997_{\pm0.000}$	$0.989_{\pm0.001}$	$0.586_{\pm0.002}$	$0.900_{\pm0.002}$	$0.979_{\pm0.001}$	$0.991_{\pm0.001}$	$0.992_{\pm0.001}$	$0.991_{\pm0.001}$
default	$0.493_{\pm0.009}$	$0.991_{\pm0.001}$	$0.998_{\pm0.001}$	$0.997_{\pm0.001}$	$0.821_{\pm0.002}$	$0.995_{\pm0.000}$	$0.902_{\pm0.001}$	$0.827_{\pm0.004}$	$0.802_{\pm0.003}$	$0.871_{\pm0.001}$
diabetes	$0.367_{\pm0.001}$	$0.854_{\pm0.002}$	$0.935_{\pm0.002}$	$0.997_{\pm0.001}$	-	-	$0.940_{\pm0.001}$	$0.858_{\pm0.001}$	$0.780_{\pm0.002}$	$0.866_{\pm0.002}$
lending	$0.926_{\pm0.004}$	$0.997_{\pm0.001}$	$0.995_{\pm0.002}$	$0.995_{\pm0.001}$	$1.000_{\pm0.000}$	-	$0.998_{\pm0.001}$	$0.955_{\pm0.006}$	$0.954_{\pm0.009}$	$0.961_{\pm0.004}$
news	$0.993_{\pm0.001}$	$0.998_{\pm0.000}$	$1.000_{\pm0.000}$	$1.000_{\pm0.000}$	$0.966_{\pm0.002}$	$1.000_{\pm0.000}$	$0.999_{\pm0.000}$	$0.973_{\pm0.001}$	$0.953_{\pm0.001}$	$0.977_{\pm0.001}$
nmes	$0.905_{\pm0.007}$	$0.987_{\pm0.002}$	$0.992_{\pm0.003}$	$0.988_{\pm0.002}$	$0.650_{\pm0.014}$	$0.988_{\pm0.000}$	$0.841_{\pm0.008}$	$0.636_{\pm0.008}$	$0.623_{\pm0.008}$	$0.642_{\pm0.010}$

Table 16: Distance to closest record of the generated data (incl. standard errors in subscripts) for seven benchmark models and for CDTD with three different noise schedules.

	Test Set	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
acsincome	$7.673_{\pm0.017}$	-	$8.637_{\pm0.027}$	$10.758_{\pm0.054}$	$6.652_{\pm0.032}$	-	$10.877_{\pm0.092}$	$10.305_{\pm0.073}$	$8.346_{\pm0.056}$	$8.322_{\pm0.047}$	$8.349_{\pm0.033}$
adult	$1.870_{\pm0.000}$	$1.371_{\pm0.018}$	$2.523_{\pm0.012}$	$5.012_{\pm0.028}$	$2.227_{\pm0.013}$	$1.647_{\pm0.009}$	$2.735_{\pm0.028}$	$2.341_{\pm0.013}$	$1.112_{\pm0.019}$	$1.231_{\pm0.011}$	$1.294_{\pm0.009}$
bank	$2.369_{\pm0.000}$	$1.369_{\pm0.011}$	$3.025_{\pm0.017}$	$3.840_{\pm0.014}$	$3.136_{\pm0.007}$	$2.327_{\pm0.010}$	$3.062_{\pm0.012}$	$2.973_{\pm0.012}$	$1.828_{\pm0.008}$	$1.943_{\pm0.007}$	$2.062_{\pm0.008}$
beijing	$0.385_{\pm0.000}$	$0.139_{\pm0.003}$	$0.735_{\pm0.003}$	$1.004_{\pm0.006}$	$0.926_{\pm0.003}$	$0.739_{\pm0.006}$	$0.610_{\pm0.002}$	$0.626_{\pm0.001}$	$0.490_{\pm0.002}$	$0.489_{\pm0.001}$	$0.477_{\pm0.002}$
churn	$0.347_{\pm0.000}$	$0.232_{\pm0.028}$	$1.136_{\pm0.015}$	$1.804_{\pm0.036}$	$1.146_{\pm0.039}$	$0.342_{\pm0.031}$	$0.852_{\pm0.016}$	$1.130_{\pm0.018}$	$0.332_{\pm0.021}$	$0.274_{\pm0.021}$	$0.276_{\pm0.012}$
covertype	$0.529_{\pm0.001}$	-	$1.741_{\pm0.011}$	$5.773_{\pm0.017}$	$3.173_{\pm0.013}$	$0.889_{\pm0.007}$	$1.508_{\pm0.020}$	$3.086_{\pm0.009}$	$2.297_{\pm0.026}$	$2.209_{\pm0.022}$	$2.252_{\pm0.013}$
default	$1.812_{\pm0.000}$	$1.032_{\pm0.010}$	$3.095_{\pm0.026}$	$5.880_{\pm0.020}$	$3.216_{\pm0.013}$	$1.422_{\pm0.013}$	$2.593_{\pm0.020}$	$2.603_{\pm0.018}$	$1.127_{\pm0.028}$	$1.269_{\pm0.014}$	$1.253_{\pm0.012}$
diabetes	$15.608_{\pm0.055}$	$13.909_{\pm0.050}$	$17.736_{\pm0.107}$	$21.935_{\pm0.046}$	$8.214_{\pm0.022}$	-	-	$28.955_{\pm0.060}$	$15.279_{\pm0.026}$	$15.126_{\pm0.058}$	$15.350_{\pm0.059}$
lending	$11.184_{\pm0.000}$	$17.752_{\pm0.143}$	$17.776_{\pm0.132}$	$20.239_{\pm0.222}$	$10.688_{\pm0.025}$	$12.537_{\pm0.076}$	-	$16.222_{\pm0.092}$	$13.775_{\pm0.147}$	$14.162_{\pm0.188}$	$13.966_{\pm0.282}$
news	$3.615_{\pm0.000}$	$3.553_{\pm0.134}$	$6.147_{\pm0.010}$	$4.789_{\pm0.005}$	$5.821_{\pm0.003}$	$4.960_{\pm0.006}$	$4.661_{\pm0.023}$	$5.351_{\pm0.008}$	$3.635_{\pm0.004}$	$3.687_{\pm0.006}$	$3.749_{\pm0.048}$
nmes	$1.931_{\pm0.000}$	$1.394_{\pm0.019}$	$2.203_{\pm0.028}$	$2.971_{\pm0.008}$	$1.710_{\pm0.019}$	$0.891_{\pm0.033}$	$1.231_{\pm0.024}$	$2.260_{\pm0.034}$	$0.664_{\pm0.029}$	$0.710_{\pm0.032}$	$0.771_{\pm0.023}$

Table 17: Machine learning efficiency F1 score for seven benchmark models, the real training data and for CDTD with three different noise schedules. The standard deviation takes into account five different sampling seeds and uses the average results of the four machine learning efficiency models computed across ten model seeds.

	Real Data	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
adult	$0.797_{\pm0.000}$	$0.784_{\pm0.001}$	$0.769_{\pm0.002}$	$0.647_{\pm0.015}$	$0.756_{\pm0.002}$	$0.787_{\pm0.001}$	$0.745_{\pm0.004}$	$0.782_{\pm0.001}$	$0.787_{\pm0.001}$	$0.787_{\pm0.001}$	$0.787_{\pm0.001}$
bank	$0.745_{\pm0.002}$	$0.740_{\pm0.004}$	$0.682_{\pm0.006}$	$0.680_{\pm0.006}$	$0.629_{\pm0.006}$	$0.720_{\pm0.006}$	$0.673_{\pm0.006}$	$0.711_{\pm0.007}$	$0.776_{\pm0.003}$	$0.767_{\pm0.004}$	$0.737_{\pm0.004}$
churn	$0.873_{\pm0.003}$	$0.865_{\pm0.008}$	$0.780_{\pm0.015}$	$0.761_{\pm0.009}$	$0.802_{\pm0.017}$	$0.857_{\pm0.007}$	$0.865_{\pm0.008}$	$0.771_{\pm0.014}$	$0.854_{\pm0.011}$	$0.852_{\pm0.006}$	$0.845_{\pm0.011}$
covertype	$0.817_{\pm0.001}$	-	$0.783_{\pm0.001}$	$0.442_{\pm0.008}$	$0.711_{\pm0.002}$	$0.799_{\pm0.001}$	$0.767_{\pm0.001}$	$0.614_{\pm0.015}$	$0.734_{\pm0.002}$	$0.754_{\pm0.001}$	$0.722_{\pm0.002}$
default	$0.674_{\pm0.001}$	$0.677_{\pm0.001}$	$0.627_{\pm0.003}$	$0.686_{\pm0.002}$	$0.632_{\pm0.007}$	$0.678_{\pm0.002}$	$0.638_{\pm0.008}$	$0.496_{\pm0.009}$	$0.670_{\pm0.002}$	$0.671_{\pm0.001}$	$0.673_{\pm0.003}$
diabetes	$0.621_{\pm0.002}$	$0.615_{\pm0.002}$	$0.572_{\pm0.005}$	$0.557_{\pm0.004}$	$0.553_{\pm0.003}$	-	-	$0.560_{\pm0.006}$	$0.617_{\pm0.002}$	$0.617_{\pm0.002}$	$0.611_{\pm0.002}$

Table 18: Machine learning efficiency AUC score for seven benchmark models, the real training data and for CDTD with three different noise schedules. The standard deviation takes into account five different sampling seeds and uses the average results of the four machine learning efficiency models computed across ten model seeds.

	Real Data	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
adult	$0.915_{\pm0.000}$	$0.906_{\pm0.001}$	$0.901_{\pm0.000}$	$0.836_{\pm0.006}$	$0.889_{\pm0.002}$	$0.908_{\pm0.000}$	$0.880_{\pm0.005}$	$0.906_{\pm0.001}$	$0.910_{\pm0.000}$	$0.910_{\pm0.001}$	$0.909_{\pm0.000}$
bank	$0.947_{\pm0.000}$	$0.943_{\pm0.001}$	$0.938_{\pm0.001}$	$0.934_{\pm0.003}$	$0.830_{\pm0.020}$	$0.940_{\pm0.005}$	$0.929_{\pm0.005}$	$0.939_{\pm0.003}$	$0.945_{\pm0.000}$	$0.945_{\pm0.001}$	$0.943_{\pm0.004}$
churn	$0.964_{\pm0.001}$	$0.961_{\pm0.002}$	$0.939_{\pm0.007}$	$0.882_{\pm0.006}$	$0.948_{\pm0.004}$	$0.957_{\pm0.004}$	$0.961_{\pm0.001}$	$0.919_{\pm0.006}$	$0.962_{\pm0.001}$	$0.962_{\pm0.001}$	$0.959_{\pm0.003}$
covertype	$0.892_{\pm0.000}$	-	$0.860_{\pm0.001}$	$0.677_{\pm0.007}$	$0.777_{\pm0.001}$	$0.876_{\pm0.000}$	$0.845_{\pm0.001}$	$0.671_{\pm0.013}$	$0.816_{\pm0.002}$	$0.828_{\pm0.001}$	$0.802_{\pm0.002}$
default	$0.768_{\pm0.000}$	$0.759_{\pm0.003}$	$0.754_{\pm0.002}$	$0.744_{\pm0.002}$	$0.751_{\pm0.004}$	$0.763_{\pm0.002}$	$0.739_{\pm0.008}$	$0.746_{\pm0.011}$	$0.762_{\pm0.003}$	$0.765_{\pm0.002}$	$0.765_{\pm0.002}$
diabetes	$0.693_{\pm0.001}$	$0.679_{\pm0.001}$	$0.669_{\pm0.002}$	$0.626_{\pm0.003}$	$0.592_{\pm0.002}$	-	-	$0.645_{\pm0.002}$	$0.675_{\pm0.001}$	$0.673_{\pm0.001}$	$0.667_{\pm0.001}$

Table 19: Machine learning efficiency RMSE for seven benchmark models, the real training data and for CDTD with three different noise schedules. The standard deviation takes into account five different sampling seeds and uses the average results of the four machine learning efficiency models computed across ten model seeds.

	Real Data	SMOTE	ARF	CTGAN	TVAE	TabDDPM	CoDi	TabSyn	CDTD (single)	CDTD (per type)	CDTD (per feature)
acsincome	$0.804_{\pm0.012}$	-	$0.757_{\pm0.007}$	$2.292_{\pm0.013}$	$1.054_{\pm0.011}$	-	$0.857_{\pm0.010}$	$0.959_{\pm0.012}$	$0.838_{\pm0.015}$	$0.811_{\pm0.014}$	$0.820_{\pm0.011}$
beijing	$0.712_{\pm0.001}$	$0.739_{\pm0.002}$	$0.792_{\pm0.007}$	$1.246_{\pm0.010}$	$1.690_{\pm0.016}$	$0.606_{\pm0.006}$	$0.912_{\pm0.005}$	$0.788_{\pm0.011}$	$0.774_{\pm0.005}$	$0.770_{\pm0.005}$	$0.762_{\pm0.005}$
lending	$0.030_{\pm0.000}$	$0.042_{\pm0.001}$	$0.274_{\pm0.007}$	$0.137_{\pm0.007}$	$0.404_{\pm0.007}$	$0.795_{\pm0.031}$	-	$0.268_{\pm0.004}$	$0.061_{\pm0.001}$	$0.060_{\pm0.001}$	$0.066_{\pm0.002}$
news	$1.001_{\pm0.002}$	$1.180_{\pm0.107}$	$0.923_{\pm0.052}$	$1.906_{\pm0.019}$	$3.999_{\pm0.175}$	$0.083_{\pm0.001}$	$1.302_{\pm0.074}$	$0.374_{\pm0.028}$	$0.819_{\pm0.103}$	$0.776_{\pm0.091}$	$0.755_{\pm0.066}$
nmes	$1.001_{\pm0.003}$	$1.112_{\pm0.044}$	$0.972_{\pm0.024}$	$1.331_{\pm0.052}$	$1.127_{\pm0.047}$	$1.154_{\pm0.047}$	$1.137_{\pm0.052}$	$0.535_{\pm0.013}$	$1.108_{\pm0.083}$	$1.184_{\pm0.076}$	$1.203_{\pm0.081}$

S ABLATION STUDY DETAILS

Table 20: L_2 norm (incl. standard errors in subscripts) of the correlation matrix differences of real and synthetic train sets for five CDTD configurations with progressive addition of model components.

Configuration	**A**	**B**	**C**	**D**	**CDTD (per type)**
acsincome	$0.131_{\pm 0.003}$	$0.119_{\pm 0.004}$	$0.124_{\pm 0.006}$	$0.129_{\pm 0.004}$	$0.129_{\pm 0.003}$
adult	$0.131_{\pm 0.007}$	$0.128_{\pm 0.008}$	$0.168_{\pm 0.017}$	$0.107_{\pm 0.011}$	$0.125_{\pm 0.009}$
beijing	$0.065_{\pm 0.009}$	$0.066_{\pm 0.012}$	$0.067_{\pm 0.011}$	$0.067_{\pm 0.010}$	$0.073_{\pm 0.009}$
churn	$0.244_{\pm 0.015}$	$0.272_{\pm 0.034}$	$0.299_{\pm 0.066}$	$0.264_{\pm 0.012}$	$0.289_{\pm 0.043}$

Table 21: Jensen-Shannon divergence (incl. standard errors in subscripts) for five CDTD configurations with progressive addition of model components.

Configuration	**A**	**B**	**C**	**D**	**CDTD (per type)**
acsincome	$0.025_{\pm 0.000}$	$0.025_{\pm 0.001}$	$0.025_{\pm 0.001}$	$0.024_{\pm 0.001}$	$0.024_{\pm 0.000}$
adult	$0.012_{\pm 0.001}$	$0.013_{\pm 0.000}$	$0.012_{\pm 0.000}$	$0.014_{\pm 0.001}$	$0.013_{\pm 0.001}$
beijing	$0.004_{\pm 0.001}$	$0.006_{\pm 0.002}$	$0.005_{\pm 0.003}$	$0.004_{\pm 0.002}$	$0.005_{\pm 0.002}$
churn	$0.010_{\pm 0.002}$	$0.008_{\pm 0.002}$	$0.009_{\pm 0.004}$	$0.010_{\pm 0.002}$	$0.012_{\pm 0.002}$

Table 22: Wasserstein distance (incl. standard errors in subscripts) for five CDTD configurations with progressive addition of model components.

Configuration	**A**	**B**	**C**	**D**	**CDTD (per type)**
acsincome	$0.002_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.001_{\pm 0.000}$	$0.001_{\pm 0.000}$
adult	$0.004_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.004_{\pm 0.000}$
beijing	$0.003_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.003_{\pm 0.000}$
churn	$0.006_{\pm 0.001}$	$0.006_{\pm 0.000}$	$0.006_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.007_{\pm 0.001}$

Table 23: Detection score (incl. standard errors in subscripts) for five CDTD configurations with progressive addition of model components.

Configuration	**A**	**B**	**C**	**D**	**CDTD (per type)**
acsincome	$0.534_{\pm 0.002}$	$0.534_{\pm 0.001}$	$0.538_{\pm 0.003}$	$0.532_{\pm 0.002}$	$0.532_{\pm 0.004}$
adult	$0.597_{\pm 0.002}$	$0.593_{\pm 0.001}$	$0.615_{\pm 0.003}$	$0.580_{\pm 0.003}$	$0.588_{\pm 0.002}$
beijing	$0.953_{\pm 0.002}$	$0.959_{\pm 0.001}$	$0.952_{\pm 0.003}$	$0.953_{\pm 0.001}$	$0.949_{\pm 0.001}$
churn	$0.557_{\pm 0.014}$	$0.573_{\pm 0.014}$	$0.564_{\pm 0.012}$	$0.541_{\pm 0.015}$	$0.533_{\pm 0.007}$

Table 24: Distance to closest record of the generated data (incl. standard errors in subscripts) for five CDTD configurations with progressive addition of model components.

	Real Test Set	**A**	**B**	**C**	**D**	**CDTD (per type)**
`acsincome`	$7.673_{\pm 0.017}$	$8.335_{\pm 0.064}$	$8.222_{\pm 0.035}$	$8.305_{\pm 0.021}$	$8.352_{\pm 0.025}$	$8.322_{\pm 0.047}$
`adult`	$1.870_{\pm 0.000}$	$1.221_{\pm 0.018}$	$1.294_{\pm 0.015}$	$1.252_{\pm 0.014}$	$1.427_{\pm 0.008}$	$1.231_{\pm 0.011}$
`beijing`	$0.385_{\pm 0.000}$	$0.545_{\pm 0.001}$	$0.559_{\pm 0.002}$	$0.539_{\pm 0.003}$	$0.541_{\pm 0.002}$	$0.489_{\pm 0.001}$
`churn`	$0.347_{\pm 0.000}$	$0.307_{\pm 0.016}$	$0.326_{\pm 0.009}$	$0.294_{\pm 0.022}$	$0.298_{\pm 0.013}$	$0.274_{\pm 0.021}$

Table 25: Machine learning efficiency F1 score for five CDTD configurations with progressive addition of model components. The standard deviation accounts for five different sampling seeds and uses the average results of the four machine learning efficiency models across ten model seeds.

	Real Data	**A**	**B**	**C**	**D**	**CDTD (per type)**
`adult`	$0.797_{\pm 0.000}$	$0.788_{\pm 0.001}$	$0.788_{\pm 0.001}$	$0.787_{\pm 0.001}$	$0.788_{\pm 0.002}$	$0.787_{\pm 0.001}$
`churn`	$0.873_{\pm 0.003}$	$0.856_{\pm 0.008}$	$0.856_{\pm 0.014}$	$0.857_{\pm 0.007}$	$0.849_{\pm 0.006}$	$0.852_{\pm 0.006}$

Table 26: Machine learning efficiency AUC score for five CDTD configurations with progressive addition of model components. The standard deviation accounts for five different sampling seeds and uses the average results of the four machine learning efficiency models across ten model seeds.

	Real Data	**A**	**B**	**C**	**D**	**CDTD (per type)**
`adult`	$0.915_{\pm 0.000}$	$0.909_{\pm 0.000}$	$0.910_{\pm 0.000}$	$0.909_{\pm 0.000}$	$0.910_{\pm 0.000}$	$0.910_{\pm 0.001}$
`churn`	$0.964_{\pm 0.001}$	$0.962_{\pm 0.002}$	$0.961_{\pm 0.003}$	$0.960_{\pm 0.002}$	$0.961_{\pm 0.001}$	$0.962_{\pm 0.001}$

Table 27: Machine learning efficiency RMSE for five CDTD configurations with progressive addition of model components. The standard deviation accounts for five different sampling seeds and uses the average results of the four machine learning efficiency models across ten model seeds.

	Real Data	**A**	**B**	**C**	**D**	**CDTD (per type)**
`acsincome`	$0.804_{\pm 0.012}$	$0.815_{\pm 0.009}$	$0.813_{\pm 0.018}$	$0.823_{\pm 0.017}$	$0.814_{\pm 0.014}$	$0.811_{\pm 0.014}$
`beijing`	$0.712_{\pm 0.001}$	$0.782_{\pm 0.004}$	$0.785_{\pm 0.004}$	$0.778_{\pm 0.005}$	$0.776_{\pm 0.004}$	$0.770_{\pm 0.005}$

T TRAINING AND SAMPLING TIMES DETAILS

Table 28: Training times in minutes. TabDDPM produces NaNs during training on `acsincome` and `diabetes`, and is therefore excluded for these data. CoDi is considered prohibitively expensive to train on `diabetes` and `lending` and we report estimated (est.) training times.

	SMOTE	**ARF**	**CTGAN**	**TVAE**	**TabDDPM**	**CoDi**	**TabSyn**	**CDTD (per feature)**
`acsincome`	-	80.3	59.9	26.0	-	231.9	13.4	5.8
`adult`	-	7.4	36.2	23.7	38.3	48.3	32.7	6.9
`bank`	-	11.0	37.6	24.6	40.5	42.7	48.5	26.3
`beijing`	-	3.7	34.3	23.9	36.1	24.9	25.8	23.4
`churn`	-	0.3	27.1	13.7	18.2	25.7	21.5	6.1
`covertype`	-	130.2	58.0	36.5	44.9	69.2	30.7	28.2
`default`	-	12.0	38.3	24.8	38.9	45.9	40.1	26.4
`diabetes`	-	58.5	90.1	25.3	-	870 (est.)	34.6	26.9
`lending`	-	5.2	157.9	36.6	48.7	3000 (est.)	42.1	25.3
`news`	-	23.0	48.8	33.3	37.2	41.5	57.9	25.2
`nmes`	-	0.4	32.8	17.2	24.9	30.2	31.0	6.3

Table 29: Sample times in seconds per 1000 samples.

	SMOTE	**ARF**	**CTGAN**	**TVAE**	**TabDDPM**	**CoDi**	**TabSyn**	**CDTD (per feature)**
`acsincome`	4674.45	4.20	0.23	0.07	-	10.26	3.53	0.59
`adult`	10.71	1.78	0.31	0.16	0.82	3.65	0.88	0.56
`bank`	16.19	2.24	0.44	0.44	0.87	3.38	0.80	0.64
`beijing`	3.98	0.34	0.41	0.32	2.09	2.45	0.99	0.26
`churn`	0.52	1.00	0.40	0.24	0.95	2.78	0.80	0.39
`covertype`	10913.34	9.74	0.28	0.25	2.45	4.35	0.85	1.97
`default`	10.00	2.07	0.27	0.25	0.86	3.48	0.82	0.60
`diabetes`	166.75	5.87	0.53	0.15	-	-	0.83	1.33
`lending`	4.06	2.49	0.45	0.54	4.33	-	0.85	0.69
`news`	66.49	3.89	0.43	0.30	5.13	2.93	0.86	0.85
`nmes`	0.69	1.54	0.31	0.17	4.17	2.91	0.82	0.55