Continuous Diffusion for Mixed-Type Tabular Data

Markus Mueller **Kathrin Gruber** **Dennis Fok**
Econometric Institute
Erasmus University Rotterdam
{mueller, gruber, dfok}@ese.eur.nl

Abstract

Score-based generative models or diffusion models have proven successful across many domains in generating texts and images. However, the consideration of mixed-type tabular data with this model family has fallen short so far. Existing research mainly combines continuous and categorical diffusion processes and does not explicitly account for the feature heterogeneity inherent to tabular data. In this paper, we combine score matching and score interpolation to ensure a common type of continuous noise distribution that affects both continuous and categorical features. Further, we investigate the impact of distinct noise schedules per feature or per data type. We allow for adaptive, learnable noise schedules to ensure optimally allocated model capacity and balanced generative capability. Results show that our model outperforms the benchmark models consistently and that accounting for heterogeneity within the noise schedule design boosts sample quality.

1 Introduction

Score-based generative models [1], also known as diffusion models [2, 3], have demonstrated outstanding capabilities for the generation of images [4, 5], videos [6], text [7–9], molecules [10] and many other highly complex data structures. Although their standard formulation only applies to continuous data, the framework has since been adapted to categorical data in various ways, including discrete diffusion processes [11, 12], diffusion in continuous embedding space [7, 8, 13, 14] or other approaches [15–17]. However, adaptions to mixed-type tabular data, which includes both continuous and categorical features simultaneously, are lagging behind.

A crucial component in score-based generative models is the noise schedule [9, 18–21]. Typical noise schedule designs try to focus learning on the timesteps most important to obtaining high quality samples. Others attempt to learn the optimal noise schedule [8, 18]. Existing approaches combine distinct diffusion processes for continuous and discrete data to derive a joint model for mixed-type data [22, 23] or treat one-hot encoded categorical features as continuous during training [24]. However, the inherently different types of diffusion processes make it difficult to optimally balance the noise schedules across features (and feature types) which in turn negatively affects the model's capacity allocation across timesteps. On the other hand, non-continuous noise processes do not allow the application of accelerated sampling [25] or classifier-free guidance [26], state-of-the-art techniques developed in the image domain. Most importantly, the domain, nature and marginal distribution of features in mixed-type tabular data can vary drastically [27]. For instance, any two continuous features may be subject to different levels of discretization or different bounds, even after applying common pre-processing techniques. Any two categorical features may have different categories associated with them or one is much more imbalanced than the other. Therefore, an effective modeling of the joint distribution of mixed-type tabular data with a diffusion model warrants potentially different noise schedules per feature or data type.

In this paper, we investigate possibilities for accounting for the high feature heterogeneity in tabular data. First, we combine score matching [28] with the recently proposed score interpolation method

NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI.

[8] to derive a score-based model for mixed-type data that uses a Gaussian diffusion process for both continuous and categorical features. This way, the noise processes become directly comparable and easier to balance against each other. Second, instead of a single noise schedule across all features, we investigate the use of feature-specific noise schedules, distinct noise schedules per feature type, a single noise schedule for both types, and a single noise schedule for continuous features while imposing feature-specific noise schedules on categorical features. Lastly, we make those noise schedules adaptive such that the noise schedule can directly take feature or type heterogeneity into account. This ensures a better allocation of model capacity across features and timesteps both during training and generation and ensures high quality samples.

2 Model Preliminaries

First, we briefly explain the score-based frameworks for continuous and categorical data types. These are afterwards combined in a single diffusion model to learn the joint distribution of the data.

2.1 Score-based Generative Model for Continuous Features

Let $\{\mathbf{x}_t\}_{t=0}^{T}$ be a diffusion process that gradually adds noise in continuous time $t \in [0, T]$ to $\mathbf{x}_0 \equiv \mathbf{x}_{\text{cont}} \in \mathbb{R}^{K_{\text{cont}}}$, the vector of continuous features. The data sample distribution at time t, $p_t(\mathbf{x})$, evolves from the real data distribution $p_0(\mathbf{x})$ to a terminal distribution $p_T(\mathbf{x})$. Our goal is to learn the reverse process that allows us to go from noise $\mathbf{x}_T \sim p_T(\mathbf{x})$ to a new data sample $\mathbf{x}_0^* \sim p_0(\mathbf{x})$.

The forward-pass of such a 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 the Brownian motion [1]. The reversion of this diffusion process yields the trajectory of $\mathbf{x}$ as t goes backwards in time from T to 0. This reverse-time process can be formulated as a probability flow ordinary differential equation (ODE):

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

[see also 1]. In Eq. (2), the only unknown is the score function $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$. We approximate the score function by training a time-dependent score-based model $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)$, parameterized by $\boldsymbol{\theta}$, via *score matching* [28]: The denoising score matching (DSM) objective is

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

where $\lambda : [0, T] \to \mathbb{R}_+$ is a positive weighting function for timesteps t, $t \sim \mathcal{U}_{[0,T]}$, and $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ is the noise-inducing conditional distribution that adds noise to the ground-truth data [29].

An affine function for $\mathbf{f}(\cdot, t)$ implies $p_{0t}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_0, \sigma_t^2 I)$, and with T being sufficiently large the reverse process is started by sampling $\mathbf{x}_T \sim p_T(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma_T^2 I)$. We then guide data samples towards high density regions in the data space, for each possible timestep in the reverse process, with the trained score model as a replacement for the true and unknown score function. For this iterative denoising process, blackbox ODE or predictor-corrector samplers can be used [1].

2.2 Score-based Generative Model for Categorical Features

Since the score function is undefined for categorical data, the score-based generative framework cannot be directly applied to such data. Dieleman et al. [8] propose *score interpolation* to push the diffusion of categorical data into Euclidean embedding space. This way, unlike other diffusion models for mixed-type data, we can impose the same type of noise distribution on both categorical and continuous features. The model is able to take feature-specific uncertainty at intermediate timesteps fully into account, which improves the consistency of generated samples [8]. We show that score interpolation also facilitates an efficient modeling of mixed-type data as more subtle dependencies across types can be captured.

Let $x_{\text{cat}}^{(j)} \in \{1, \dots, C_j\}$ be the jth categorical feature with C_j possible classes. We can associate a distinct, trainable d-dimensional embedding vector $\mathbf{e}_i \in \mathbb{R}^d$ with each class $i = 1, \dots, C_j$ We denote the embedding vector corresponding to the ground truth class for a single feature as $\mathbf{x}_0 \in \{\mathbf{e}_1, \dots, \mathbf{e}_{C_j}\}$ and its noisy variant at time t as $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_0, \sigma_t^2 I_d)$.

Given $\mathbf{x}_t$ and t, the expectation $\mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,t)}[\nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t|\mathbf{x}_0)]$ is the minimizer of Eq. (3). Accordingly, we have

$$\mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,t)} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0) = \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,t)} \frac{\mathbf{x}_0 - \mathbf{x}_t}{\sigma_t^2} = \frac{\mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,t)}[\mathbf{x}_0] - \mathbf{x}_t}{\sigma_t^2}. \tag{4}$$

Thus, we can train a model to estimate $p(\mathbf{x}_0|\mathbf{x}_t, t)$ and obtain $\hat{\mathbf{x}}_0 = \mathbb{E}_{p(\mathbf{x}_0|\mathbf{x}_t,t)}[\mathbf{x}_0]$ as the weighted average over the C_j possible embedding vectors, to derive a score function estimate. Since $p(\mathbf{x}_0 = \mathbf{e}_i|\mathbf{x}_t, t) = p(x_{\text{cat}}^{(j)} = i|\mathbf{x}_t, t)$, an estimate of $p(\mathbf{x}_0|\mathbf{x}_t, t)$ can be obtained via a classifier that predicts C_j class probabilities for the jth feature and is trained via cross-entropy loss. The same framework applies to all categorical features in the dataset.

We start the generative process for each categorical feature from an embedding vector $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 I_d)$. We then use the learned classifier and score interpolation to gradually denoise the embedding vectors. After the last timestep, we use the predicted feature-specific class probabilities to directly infer the generated classes.

3 Continuous Diffusion for Mixed-Type Tabular Data

In order to learn the joint distribution of tabular data with continuous and categorical features, we combine the *score matching* (see Eq. (3)) and *score interpolation* (see Eq. (4)) to retrieve the score function estimates $\hat{s}_{\text{cont}}^{(i)}$ and $\hat{s}_{\text{cat}}^{(j)}$, respectively. An overview of our Continuous Diffusion for Mixed-Type Tabular Data (CDTD) framework is given in Figure 1. The Gaussian noise process acts directly on the continuous features, but on the embeddings of categorical features. This ensures a common *continuous* noise process for both data types, and also enables the application of, for instance, classifier-free guidance [26], to both types of features. The noise processes are directly influenced by potentially feature-specific transformed timesteps t_i, t_j, derived from timewarping. We use a joint Transformer model to parameterize the score model and the classifier required for score interpolation, and to allow for detailed across-type inter-dependencies. We adapt the Diffusion Transformer [30] to tabular data by embedding continuous features using sinusoidal embeddings. The implementation details are given in Appendix E.

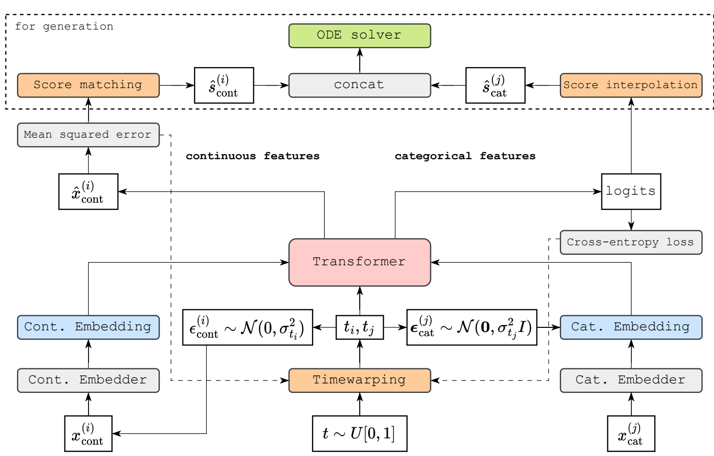

Figure 1: Continuous Diffusion for Mixed-Type Tabular Data (CDTD). The left side relates to the score model for continuous features, the right side describes the score interpolation for categorical features. Timewarping allows for a potentially feature-specific, learnable timestep, to diffuse the scalar values (for continuous features) or the embeddings (for categorical features). The approximated score functions are concatenated and passed to a blackbox ODE solver during the sample generation.

For continuous features, the Transformer output is used to predict the ground-truth scalar value; for categorical features, it yields the probability of each class. The model inputs are the set of noisy scalar values and noisy embeddings for continuous and categorical features, respectively. We also condition the model on the potentially feature- or type-specific timesteps, t_k, which affect the respective noise schedules and are derived from the timewarping we discuss below.

We train the model on the joint loss function:

$$\mathcal{L}_{\text{joint}}(\boldsymbol{\theta}) = \frac{1}{\alpha K_{\text{cat}} + (1-\alpha) K_{\text{cont}}} \Big[(1-\alpha) \sum_{i=1}^{K_{\text{cont}}} \ell_{\text{cont}}^{(i)}(\boldsymbol{\theta}) + \alpha \sum_{j=1}^{K_{\text{cat}}} \ell_{\text{cat}}^{(j)}(\boldsymbol{\theta}) \Big], \tag{5}$$

where $\ell_{\text{cont}}^{(i)}(\boldsymbol{\theta})$ is the score matching loss of the ith continuous feature, $\ell_{\text{cat}}^{(j)}(\boldsymbol{\theta})$ is the cross-entropy loss of the jth categorical feature, and α is the relative weight of the two loss types, which for simplicity we set to 0.5 hereafter. For sampling, the concatenated score function estimates are input to an ODE solver (Euler with 200 steps to minimize the discretization error).

3.1 Feature-specific and Adaptive Noise Schedules

Even though we embed all categorical features in the same Euclidean embedding space, it is unlikely that a single noise schedule is optimal for all features. For a given embedding dimension, more noise is needed to remove a given amount of signal from embeddings of features with fewer categories. Also, continuous features have different domains and distributions, so that a single common noise schedule may not be optimal for those features either.

As a solution, we investigate the use of feature-specific noise schedules, distinct noise schedules per data type, a single noise schedule for both types, and lastly a mixture of a single noise schedule for continuous features and feature-specific noise schedules for categorical ones. In the image domain a similar idea was coined non-uniform diffusion [31], which involves diffusing different groups of pixels at different speeds. For brevity, we introduce only the feature-specific noise schedules and timewarping explicitly. The other types of noise schedules we investigate are easily derived from the feature-specific variant by appropriately combining terms.

We let the ith continuous feature follow the diffusion process given by

$$\mathrm{d}x_{\text{cont}}^{(i)} = f_{\text{cont},i}(x_{\text{cont}}^{(i)}, t)\mathrm{d}t + g_{\text{cont},i}(t)\mathrm{d}w_t^{(i)}, \tag{6}$$

and describe the trajectory of the embedding of the jth categorical feature as

$$\mathrm{d}\mathbf{x}_{\text{cat}}^{(j)} = \mathbf{f}_{\text{cat},j}(\mathbf{x}_{\text{cat}}^{(j)}, t)\mathrm{d}t + g_{\text{cat},j}(t)\mathrm{d}\mathbf{w}_t^{(j)}, \tag{7}$$

where $\mathbf{x}_{\text{cat}}^{(j)}$ represents the d-dimensional embedding of $x_{\text{cat}}^{(j)}$ in Euclidean space.

Further, we specify the feature-specific timesteps, $t_{\text{cont},i}(t), t_{\text{cat},j}(t)$ as a function of the global time t. Following Karras et al. [32], we set the drift coefficients $f_{\text{cont},i}(x_{\text{cont}}^{(i)}, t)$ and $\mathbf{f}_{\text{cat},j}(\mathbf{x}_{\text{cat}}^{(j)}, t)$ to zero and the feature-specific diffusion coefficients to $g_{\text{cont},i}(t) = \sqrt{2t_{\text{cont},i}(t)}$ and $g_{\text{cat},j}(t) = \sqrt{2t_{\text{cat},j}(t)}$. This specification implies the *feature-specific* noise distributions $x_{\text{cont},t}^{(i)} \sim \mathcal{N}(x_{\text{cont},0}^{(i)}, \sigma_{\text{cont},i,t}^2)$ and $\mathbf{x}_{\text{cat},t}^{(j)} \sim \mathcal{N}(\mathbf{x}_{\text{cat},0}^{(j)}, \sigma_{\text{cat},j,t}^2 I_d)$, with the feature-specific standard deviations (aka noise levels) $\sigma_{\text{cont},i,t} = t_{\text{cont},i}(t)$ and $\sigma_{\text{cat},j,t} = t_{\text{cat},j}(t)$. Thus, each feature is governed by a distinct noise schedule, while all elements of an embedding representing a given categorical feature follow the same noise schedule.

Typically, the functions $t_{\text{cont},i}(t)$ and $t_{\text{cat},j}(t)$ are chosen to be identity functions. However, we parameterize the functions with the active learning strategy called timewarping [8]. For each feature-specific timestep $t_k(t) \in \{t_{\text{cat},1}(t), \ldots, t_{\text{cat},K_{\text{cat}}}(t), t_{\text{cont},1}(t), \ldots, t_{\text{cont},K_{\text{cont}}}(t)\}$, we learn a non-linear map such that the learned noise schedule is optimal for both training and generation. The goal is a generative reverse process that improves the sample quality of each feature linearly in uniform time, t. Hence, model capacity is optimally allocated and benefits all features simultaneously.

Without loss of generality, we let $t \in [0, 1]$. We aim to construct a feature-specific non-linear transformation $t_k(t)$, for each feature $k \in \{1, 2, ..., K\}$, that maps the timestep, t, to a feature-specific timestep, t_k. Alongside our score model we learn K monotonic piece-wise linear functions $F_k : t_k \mapsto \ell_k$, by predicting the feature-specific denoising loss ℓ_k based on the feature-specific timesteps, t_k. Let $\tilde{F}_i$ represent the normalized version of F_i such that $\tilde{F}_k : t_k \mapsto t$, then $\tilde{F}_k^{-1}$ achieves our transformation of interest and we let $t_k(t) = \tilde{F}_k^{-1}(t)$. We apply this parameterization both during training and generation. For more details on the setup and training of the piece-linear function see the appendix in Dieleman et al. [8].

Since in the forward process of our model, we add noise directly to continuous features but to the embedding of categorical features, we generally need much more noise to remove all signal from the categorical data representations. Thus, in practice we define type-specific minimum and maximum noise levels to be $t_{\text{cat,min}} = 0.1$ and $t_{\text{cat,max}} = 200$ and follow Karras et al. [32] by setting $t_{\text{cont,min}} = 0.002$ and $t_{\text{cont,max}} = 80$. For continuous features, we use the preconditioning and weighting proposed by Karras et al. [32].

4 Experiments

We benchmark our model against several popular generative models and across multiple datasets.

Baseline models We benchmark our model against a multitude of different generative models for mixed-type tabular data. This includes SMOTE [33], TVAE [27], CTGAN [27], ARF [34], and TabDDPM [22]. All models follow a different design and / or modeling philosophy. For more details see Appendix A.

Datasets To systematically investigate our model, we consider 3 different datasets (adult, churn, nmes). These vary in size (between 3 150 and 48 842 observations), prediction task (regression vs. binary classification), number of features and their distributions. Details on the datasets are given in Appendix D. Rows with missings in either the target or any continuous feature are removed, missings in categorical features are encoded as a separate category. All datasets are split in 60% train, 20% validation, and 20% test partitions using stratification with respect to the outcome in case of a classification task. For our own model, we use a quantile transformation on the continuous features, for other models we adhere to the required respective pre-processing. For classification tasks, we condition the model the binary outcome,

Tuning framework For hyperparameter tuning on a common objective, we first tune a catboost model on the data-specific prediction task. The tuned model is then used to tune the generative model by estimating the machine learning efficiency (see below) on a validation set using data sampled by the generative model. See Appendix B for more details. For time reasons, we only tune the hyperparameters of the CDTD model with a single noise schedule for continuous features and feature-specific noise schedules for categorical features (single cont.) and use the tuned parameters for all model variants. For all models, we round integer-valued continuous features.

4.1 Evaluation Metrics

We evaluate generative models using four criteria. All metrics are averaged over five random seeds for the generative process.

Machine learning efficiency As many previous papers [22, 24, 27, 34–36], our main metric is the machine learning efficiency. A group of models consisting of a (logistic) regression, a tree, a random forest and a (tuned) catboost model are trained on the data-specific prediction task. The real test set performance of the models is compared when trained on the real training set or a synthetic training set of equal size. For regression tasks we report the MSE, for classification tasks the macro-averaged F1 score. Results are averaged over ten different model seeds and five different sampling seeds. Hyperparameters for the machine learning efficiency models are reported in Appendix C.

Statistical similarity To evaluate the statistical similarity between real and synthetic training data, we use (1) the Jensen-Shannon divergence (JSD) [37] to quantify the difference in categorical distributions, (2) the Wasserstein distance (WD) [38] to quantify the difference in continuous distributions, and (3) the L_2 norm of the pair-wise differences of the correlation matrices. For correlations between two continuous features we use the Pearson correlation coefficient, for correlations between two categorical features the Theil uncertainty coefficient and across types the correlation ratio [22, 39].

Detection score We report the accuracy of a catboost model [40] that is trained to distinguish between real and generated (fake) samples [35, 36]. Train and evaluation sets contain equal proportions of

Table 1: Evaluation of the generative models. Bold indicates the best performing model. The best results among different variants of our CDTD model are underlined.

	dataset	adult	churn	nmes
ML efficiency (↑ for adult and churn (F1), ↓ for nmes (MSE))	Original train set	$0.794_{\pm 0.016}$	$0.880_{\pm 0.074}$	$11.413_{\pm 17.631}$
	ARF	$0.769_{\pm 0.010}$	$0.789_{\pm 0.045}$	$\mathbf{11.792}_{\pm \mathbf{18.540}}$
	CTGAN	$0.765_{\pm 0.015}$	$0.735_{\pm 0.021}$	$13.800_{\pm 22.203}$
	TVAE	$0.769_{\pm 0.014}$	$0.792_{\pm 0.021}$	$13.171_{\pm 18.083}$
	SMOTE	$0.781_{\pm 0.011}$	$\mathbf{0.866}_{\pm \mathbf{0.065}}$	$12.231_{\pm 18.149}$
	TabDDPM	$0.778_{\pm 0.010}$	$0.490_{\pm 0.054}$	$18.784_{\pm 32.171}$
	CDTD (single)	$0.784_{\pm 0.009}$	$0,811_{\pm 0.040}$	$12.080_{\pm 18.446}$
	CDTD (per type)	$0.786_{\pm 0.011}$	$\underline{0.824_{\pm 0.047}}$	$12.036_{\pm 18.316}$
	CDTD (single cont.)	$\underline{\mathbf{0.787}_{\pm \mathbf{0.012}}}$	$0.816_{\pm 0.043}$	$\underline{12.000_{\pm 18.373}}$
	CDTD (per feature)	$0.780_{\pm 0.012}$	$0.821_{\pm 0.045}$	$12.021_{\pm 18.408}$
L_2 distance of correlation matrices (↓)	ARF	$0.585_{\pm 0.006}$	$0.602_{\pm 0.031}$	$0.669_{\pm 0.030}$
	CTGAN	$0.499_{\pm 0.012}$	$2.678_{\pm 0.047}$	$1.390_{\pm 0.023}$
	TVAE	$0.632_{\pm 0.009}$	$0.753_{\pm 0.025}$	$2.317_{\pm 0.070}$
	SMOTE	$0.503_{\pm 0.012}$	$\mathbf{0.283}_{\pm \mathbf{0.040}}$	$0.658_{\pm 0.042}$
	TabDDPM	$0.227_{\pm 0.029}$	$4.942_{\pm 0.042}$	$3.305_{\pm 0.053}$
	CDTD (single)	$0.104_{\pm 0.010}$	$0.475_{\pm 0.072}$	$0.588_{\pm 0.027}$
	CDTD (per type)	$\underline{\mathbf{0.093}_{\pm \mathbf{0.010}}}$	$\underline{0.441_{\pm 0.070}}$	$0.557_{\pm 0.025}$
	CDTD (single cont.)	$0.098_{\pm 0.016}$	$0.498_{\pm 0.078}$	$0.544_{\pm 0.038}$
	CDTD (per feature)	$0.125_{\pm 0.009}$	$0.514_{\pm 0.075}$	$\underline{\mathbf{0.543}_{\pm \mathbf{0.038}}}$
Detection accuracy (the closer to 0.5 the better)	ARF	$0.918_{\pm 0.003}$	$0.847_{\pm 0.006}$	$0.986_{\pm 0.003}$
	CTGAN	$0.988_{\pm 0.001}$	$0.977_{\pm 0.006}$	$0.992_{\pm 0.003}$
	TVAE	$0.931_{\pm 0.002}$	$0.915_{\pm 0.009}$	$0.990_{\pm 0.002}$
	SMOTE	$0.337_{\pm 0.003}$	$0.339_{\pm 0.019}$	$0.868_{\pm 0.009}$
	TabDDPM	$0.600_{\pm 0.002}$	$0.998_{\pm 0.002}$	$0.998_{\pm 0.001}$
	CDTD (single)	$0.561_{\pm 0.002}$	$0.850_{\pm 0.006}$	$\underline{\mathbf{0.647}_{\pm \mathbf{0.013}}}$
	CDTD (per type)	$0.559_{\pm 0.004}$	$\underline{\mathbf{0.832}_{\pm \mathbf{0.004}}}$	$0.653_{\pm 0.015}$
	CDTD (single cont.)	$\underline{\mathbf{0.557}_{\pm \mathbf{0.003}}}$	$0.847_{\pm 0.009}$	$0.649_{\pm 0.016}$
	CDTD (per feature)	$0.583_{\pm 0.002}$	$0.849_{\pm 0.009}$	$0.652_{\pm 0.011}$

real and fake samples. For each generative model, we tune a catboost model on a validation set and report the accuracy of the best-fitting model on a test set. See Appendix B for more details.

Distance to closest record (DCR) To check whether the model does not copy training samples, we check the DCR [35, 39]. First, we one-hot encode categorical features. All features are standardized to have a zero mean and unit variance to ensure that each feature contributes equally to the distance. The DCR of a given generated data point is then the minimum Euclidean distance of that data point to all observations in the true training set. As a robust estimate, we report the average DCR.

4.2 Results

Table 1 shows the results of evaluating the generative models. For our own model (CDTD), to account for feature heterogeneity we compare the effect of more vs. less feature-specific noise schedules. We compare a *single* schedule, one schedule *per type*, one schedule *per feature* and having one schedule for continuous features and feature-specific schedules for categorical features (*single cont.*). Additional results on the statistical similarity measures and DCR are given in Appendix F.

Our model consistently outperforms the benchmark models in most metrics, often substantially so. Also, explicitly accounting for heterogeneity in the specification of the noise schedules seems valuable. Even though feature-specific noise schedules appear to be too extreme and often perform worse, introducing type- or feature-specific schedules for the categorical features boosts the performance considerably. The outstanding results in terms of the L_2 distance of the correlation matrices show that a common noise distribution type across data types and the possibility to capture subtle uncertainty in the categorical features together with the transformer-backbone benefit sample quality.

References

[1] 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.

[2] 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.

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

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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*, pages 8867–8887, Baltimore, Maryland, USA, 2022. PMLR.

[11] 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.

[12] 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, pages 12454–12465. Curran Associates, Inc., 2021.

[13] 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.

[14] 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.

[15] 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.

[16] 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, pages 34532–34545. Curran Associates, Inc., 2022.

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

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

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

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

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

[22] 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*, pages 17564–17579. PMLR, 2023.

[23] 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.

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

[25] 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.

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

[27] 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.

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

[29] 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.

[30] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. *arXiv preprint arXiv:2212.09748*, 2023.

[31] Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Non-Uniform Diffusion Models. *arXiv preprint arXiv:2207.09786*, 2022.

[32] 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, pages 26565–26577. Curran Associates, Inc., 2022.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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, pages 97–112. PMLR, 2021.

[40] 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.

[41] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery.

A Benchmark Models

We benchmark our model against a diverse set of generative models that were especially built for modelling mixed-type tabular data.

- SMOTE [33] – a naive data augmentation technique based on the convex combination of training samples. For regression problems data is split into two classes using the median target. We use the SMOTE-NC version that is designed for mixed-type data.
- TVAE [27] – a Variational-Autoencoder-based model for tabular data.
- CTGAN [27] – one of the most popular Generative-Adversarial-Network-based models for tabular data.
- ARF [34] – recent generative modelling approach based on using a random forest for density estimation.
- TabDDPM [22] – first diffusion-based generative model for tabular data that combines diffusion in continuous space and multinomial diffusion [12].

A.1 Hyperparameter Searchspaces

Table 2: TVAE [27] hyperparameter space; implementation available at `https://github.com/sdv-dev/CTGAN`. We tune the model for 20 trials.

Parameter	Distribution
embedding dim.	Cat([128, 256, 512])
batch size	Cat([1024, 4096])
epochs	= 1000
loss factor	Log Uniform [1, 5]
encoder dims	= [256, 256]
decoder dims	= [256, 256]
learning rate	= 1e-3

Table 3: CTGAN [27] hyperparameter space; implementation available at `https://github.com/sdv-dev/CTGAN`. We do an exhaustive search.

Parameter	Distribution
embedding dim.	Cat([128, 256, 512])
batch size	= 4096
epochs	Cat([1000, 2000, 5000])
generator dims	= [256, 256]
discriminator dims	= [256, 256]
learning rate generator	= 2e-4
learning rate discriminator	= 2e-4

Table 4: CDTD (ours) hyperparameter space. We tune the the model for 15 trials.

Parameter	Distribution
embedding dim	= 64
cat embedding σ_{init}	= 0.001
batch size	= 4096
num. train steps	= 15000
learning rate	Log Uniform [0.00001, 0.002]
depth	= 4
heads	= 8

Table 5: TabDDPM [22] hyperparameter space. We tune the model for 20 trials.

Parameter	Distribution
epochs	= 1000
num timesteps	Cat([100, 1000])
batch size	Cat([1024, 4096])
learning rate	Log Uniform [0.00001, 0.003]
num hidden layers	= 5
hidden size	= 512

Table 6: ARF [34] hyperparameter space. We do an exhaustive search.

Parameter	Distribution
δ	= 0
min. node size	= 5
max. iters	= 50
num. trees	Cat([30, 40, 50, 60, 70, 80])

Table 7: SMOTE [33] hyperparameter space. We do an exhaustive search.

Parameter	Distribution
k neighbours	Cat([5,6,7,8,9,10,11,12,13,14,15])

Table 8: Catboost [40] hyperparameter space. For estimating the machine learning efficiency, we tune it for 100 trials. When used as a detection model, we tune it for 30 trials.

Parameter	Distribution
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]

B Tuning Setup

B.1 Tuning Generative Models

We follow Kotelnikov et al. [22] and tune the hyperparameters of generative models based on how well the generated data captures the joint distribution of the real training data. This tuning procedure ensures that all model types are tuned with a common objective.

First, for a given dataset we tune a catboost model [40] that predicts the dataset-specific target, which can be continuous or binary, based on the real training set. For regression tasks we determine the fit using Mean-Squared Error, for binary or multi-class classification we use the macro-averaged F1-score. We estimate the goodness-of-fit statistics using 5 fold cross-validation based on the real training set only. For tuning we use optuna [41] and 100 trials.

Once we have selected the best-fitting catboost model for a dataset, we use that model to do the hyperparameter tuning of the generative models. For a given set of hyperparameters of a generative model, we train the model and generate a sample of the same size as the real training set. We check the machine learning efficiency of the generative model using the tuned catboost model, i.e., we use the generated data as a drop-in replacement for the real train set and train the catboost model (using the tuned dataset-specific hyperparameters) to predict the outcome in a separate, real validation set. The tuning objective is derived by averaging the machine learning efficiency over five different seeds that affect the sampling process of the generative models. We again use optuna, but the number of trials varies across the different generative models.

B.2 Tuning Detection Model

A detection model is used to test whether a statistical model can distinguish between real and generated samples. We again use a catboost model for this purpose. To tune it we use optuna with 30 trials. For each of the real train, validation and test sets we generate the same number of fake observations. Per set we then combine real and fake observations and name them $\mathcal{D}_{\text{train,detect}}$, $\mathcal{D}_{\text{val,detect}}$ and $\mathcal{D}_{\text{test,detect}}$, respectively. The catboost model is trained on $\mathcal{D}_{\text{train,detect}}$ with the task of predicting whether an observation is real or fake. We evaluate the performance and tune the catboost model in terms of accuracy on $\mathcal{D}_{\text{val,detect}}$. After tuning, the performance of the tuned model on the held-out test set, $\mathcal{D}_{\text{test,detect}}$ represents the detection score.

C Machine Learning Efficiency Models

Table 9: Hyperparameters of models used to estimate machine learning efficiency. For all models except catboost we use the implementation and default parameters (if not specified otherwise) of the scikit-learn package. For catboost the same holds for the package of the same name.

Model	Parameters
Tree	max_depth = 12
Random Forest	max_depth = 12, n_estimators = 100
Logistic or Ridge Regression	max_iter = 1000
Catboost	tuned using space defined in Table 8

D Datasets

Table 10: Overview of the experimental datasets. We count the outcome towards the respective features.

Dataset	Link	Task	Total obs.	Num. cat. features	Num. cont. features
adult	link	bin. class.	48842	9	6
churn	link	bin. class.	3150	5	9
nmes	link	regression	4406	9	10

E Implementation Details

In the Transformer, we embed the timesteps and add them to the respective feature embeddings. Following Dieleman et al. [8], it is crucial to L_2 normalize the embeddings each time we use them, to prevent a degenerate embedding space, in which embeddings are pushed further and further apart. In a conditional model, where we condition on the score model on the target feature in the dataset, we use adaptive layer norms [30]. We also use self-conditioning [19] for both types of features. For continuous features we simply condition on the predictions from the previous step, for categorical features we condition on the interpolated embedding following Dieleman et al. [8]. We use an exponential moving average on both the parameters of the score model as well as any timewarping functions. The timewarping weights are initialized such that the initial draws are uniformly distributed and we use 100 bins per piece-wise linear spline. To decrease the variance of the loss, we use the low-discrepancy sampler to sample $t \sim \mathcal{U}_{[0,1]}$ [18].

F Additional Results

Table 11: Additional experimental results. We use the Jensen-Shannon divergence (JSD) for differences in categorical distributions and the Wasserstein distance (WD) for differences in continuous distributions. Bold indicates best performance. The distance to closest record (DCR) should neither be too low nor too high but should be compared relatively to the DCR of the real test set. Bold indicates the best performing model. The best results among different variants of our CDTD model are underlined.

	dataset	adult	churn	nmes
Wasserstein Distance (↓)	ARF	$0.011_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.012_{\pm 0.001}$
	CTGAN	$0.012_{\pm 0.001}$	$0.041_{\pm 0.001}$	$0.027_{\pm 0.001}$
	TVAE	$0.012_{\pm 0.000}$	$0.014_{\pm 0.002}$	$0.016_{\pm 0.001}$
	SMOTE	$0.002_{\pm 0.000}$	$\mathbf{0.005_{\pm 0.001}}$	$\mathbf{0.005_{\pm 0.000}}$
	TabDDPM	$0.003_{\pm 0.000}$	$0.385_{\pm 0.002}$	$0.412_{\pm 0.002}$
	CDTD (single)	$0.002_{\pm 0.016}$	$0.012_{\pm 0.001}$	$\underline{0.006_{\pm 0.000}}$
	CDTD (per type)	$\underline{\mathbf{0.001_{\pm 0.000}}}$	$\underline{0.011_{\pm 0.001}}$	$\underline{0.006_{\pm 0.000}}$
	CDTD (single cont.)	$0.002_{\pm 0.000}$	$\underline{0.011_{\pm 0.001}}$	$\underline{0.006_{\pm 0.000}}$
	CDTD (per feature)	$0.004_{\pm 0.000}$	$\underline{0.011_{\pm 0.001}}$	$\underline{0.006_{\pm 0.000}}$
Jensen-Shannon Divergence (↓)	ARF	$\mathbf{0.007_{\pm 0.000}}$	$\mathbf{0.010_{\pm 0.002}}$	$\mathbf{0.008_{\pm 0.002}}$
	CTGAN	$0.101_{\pm 0.001}$	$0.094_{\pm 0.001}$	$0.098_{\pm 0.001}$
	TVAE	$0.086_{\pm 0.001}$	$0.036_{\pm 0.003}$	$0.096_{\pm 0.002}$
	SMOTE	$0.072_{\pm 0.000}$	$0.011_{\pm 0.002}$	$0.114_{\pm 0.002}$
	TabDDPM	$0.020_{\pm 0.000}$	$0.112_{\pm 0.001}$	$0.080_{\pm 0.003}$
	CDTD (single)	$\underline{0.010_{\pm 0.000}}$	$0.013_{\pm 0.003}$	$\underline{0.010_{\pm 0.002}}$
	CDTD (per type)	$0.012_{\pm 0.000}$	$\underline{0.011_{\pm 0.002}}$	$0.012_{\pm 0.002}$
	CDTD (single cont.)	$0.011_{\pm 0.001}$	$0.014_{\pm 0.002}$	$0.013_{\pm 0.002}$
	CDTD (per feature)	$0.013_{\pm 0.000}$	$0.014_{\pm 0.002}$	$0.012_{\pm 0.002}$
Avg. Distance to Closest Record	Real test set	1.87	0.347	1.970
	ARF	$2.480_{\pm 0.009}$	$1.108_{\pm 0.019}$	$2.236_{\pm 0.036}$
	CTGAN	$3.775_{\pm 0.034}$	$2.607_{\pm 0.011}$	$2.828_{\pm 0.020}$
	TVAE	$2.308_{\pm 0.018}$	$1.140_{\pm 0.019}$	$\mathbf{1.977_{\pm 0.020}}$
	SMOTE	$1.427_{\pm 0.007}$	$\mathbf{0.224_{\pm 0.018}}$	$1.945_{\pm 0.017}$
	TabDDPM	$1.931_{\pm 0.012}$	$2.800_{\pm 0.014}$	$2.881_{\pm 0.015}$
	CDTD (single)	$\mathbf{1.885_{\pm 0.011}}$	$1.014_{\pm 0.012}$	$\underline{1.954_{\pm 0.025}}$
	CDTD (per type)	$\underline{\mathbf{1.893_{\pm 0.011}}}$	$0.946_{\pm 0.018}$	$1.943_{\pm 0.012}$
	CDTD (single cont.)	$1.902_{\pm 0.012}$	$\underline{0.991_{\pm 0.011}}$	$1.948_{\pm 0.011}$
	CDTD (per feature)	$1.912_{\pm 0.018}$	$0.997_{\pm 0.014}$	$1.947_{\pm 0.012}$