DN-Flow: Driver–Navigator Structured Flow Matching for Mixed-Type Tabular Data Generation

Guangzhao Chai
School of Artificial Intelligence and Data Science
Hebei University of Technology
Tianjin, China
cgz010321@163.com

Jing Liu
School of Artificial Intelligence and Data Science
Hebei University of Technology
Tianjin, China
liujhebut@163.com

Youxi Wu*
School of Artificial Intelligence and Data Science
Hebei University of Technology
Tianjin, China
wuc567@163.com

Yan Li
School of Economics and Management
Hebei University of Technology
Tianjin, China
lywuc@163.com

Abstract

Generating high-fidelity mixed-type tabular data remains challenging because such data contain both numerical and categorical features; different columns often exhibit substantial heterogeneity in marginal distributions, sparsity patterns, and dependency structures. Although recent diffusion-based and flow-based methods have improved generation quality, most methods still rely on a unified generation framework that models all dimensions in a homogeneous manner, without explicitly accounting for the fact that different columns may differ in their suitability for strong local refinement. To address this issue, we propose a driver-navigator structured flow-matching framework (DN-Flow) for mixed-type tabular data generation. Inspired by the collaboration between the driver and navigator in rally car racing, DN-Flow explicitly decomposes the velocity field into a Driver branch (for globally stable transport) and a Navigator branch (for endpoint-aware local refinement). On top of this structured decomposition, we introduce a learnable column-wise soft selector and a dynamic correction control mechanism to jointly model column-level suitability for local refinement and the actual strength of correction injection in an end-to-end manner. Experiments on eight real-world datasets show that DN-Flow consistently outperforms strong baselines in data fidelity and downstream utility, achieves the best Shape performance on all eight datasets, and remains highly competitive on Trend and machine learning efficiency. These results suggest that structured and selective velocity correction offers an effective new paradigm for mixed-type tabular data generation. The code is publicly available at https://github.com/xmdz521-sudo/dn-flow.

1 Introduction

Tabular data are some of the most widely used and practically valuable data forms in the real world, with applications spanning finance, healthcare, e-commerce, and scientific analysis. High-quality synthetic tabular data can support privacy-preserving data sharing, data augmentation, and downstream machine learning, making tabular data generation an important problem in modern data science (Alaa et al., 2022; Qian et al., 2023). Unlike images or text, however, tabular data are inherently heterogeneous: a single table usually contains both numerical and categorical features,

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

even columns of the same type may differ substantially in terms of marginal distributions, sparsity patterns, and cross-column dependencies (Grinsztajn et al., 2022; Yan et al., 2023; Liu et al., 2025). This heterogeneity makes high-fidelity tabular generation particularly challenging. Recent years have witnessed substantial progress in this area, including GAN-based, VAE-based, autoregressive, and diffusion-based methods (Borisov et al., 2023; Liu et al., 2023a; Kim et al., 2023). Among these, diffusion models have shown especially strong performance due to their stable training and expressive generative capabilities. Nevertheless, most existing methods still rely on a largely unified de-noising or velocity-learning mechanism that models all dimensions homogeneously (Kotelnikov et al., 2023; Lee et al., 2023; Zhang et al., 2024). Whilst being effective for learning the overall generation process, such a strategy overlooks an important property of heterogeneous tabular data: different columns are not equally suitable for the same form of strong correction. This challenge was also reflected in recent tabular learning studies, which showed that heterogeneous feature interactions remain difficult, even for strong predictive models (Holzmüller et al., 2024; McElfresh et al., 2023). In tabular generation, a similar issue arises because different columns are not equally suited to the same form of strong correction. In particular, some continuous numerical columns may benefit more from endpoint-aware refinement; whereas, other columns may require different or weaker correction mechanisms. Motivated by this observation, we propose "DN-Flow", which is a structured flow-matching framework for mixed-type tabular data generation. Instead of appending a generic residual correction to a unified velocity field, DN-Flow adopts asymmetric velocity-field decomposition with explicit role specialization: the Driver provides a full-dimensional base velocity field for globally stable transport, while the Navigator predicts an endpoint state and induces route-aware correction as a dedicated local refinement module. To further adapt this structure to heterogeneous tabular data, DN-Flow introduces a two-level control mechanism for local correction: a learnable column-wise soft selector models column-level suitability for endpoint-aware refinement, while a dynamic gate modulates the actual correction strength according to the current sample state and time step. In this way, DN-Flow jointly models global transport and controlled local refinement within a unified end-to-end framework.

The main contribution aims of this paper are as follows: 1) To address the lack of structural modeling for heterogeneous mixed-type tabular generation, we propose DN-Flow, a structured flow-matching framework that reformulates generation as a collaborative process of globally stable transport and controlled local route refinement. 2) To separate global generation stability from endpoint-aware local correction, we introduce an asymmetric Driver–Navigator velocity-field decomposition, where the Driver provides a full-dimensional base velocity field and the Navigator performs route-aware local refinement induced by a predicted endpoint, rather than acting as a generic residual correction branch. 3) To explicitly model heterogeneity in correction behavior across columns and samples, we propose a two-level control mechanism consisting of a column-wise soft selector and a dynamic gate, which disentangles column-level suitability for correction from instance-dependent and time-dependent correction strength. 4) To comprehensively validate the proposed framework, we conduct experiments on eight real-world datasets. The results show that DN-Flow is highly competitive on Shape, Trend, Machine Learning Efficiency, and additional fidelity metrics, while parameter-matched single-branch baselines and systematic ablations further indicate that its gains mainly come from the proposed structural decomposition and controlled local correction mechanism, rather than merely increased parameter count or extra output heads.

2 Related Work

Tabular data generation has become an important topic in machine learning and data mining. Unlike images or text, tabular data contain both numerical and categorical features, and different columns often exhibit substantially different marginal distributions, statistical properties, and dependencies, making high-quality generation under mixed-type and heterogeneous settings particularly challenging (Grinsztajn et al., 2022; Van Breugel & Van Der Schaar, 2024). Early methods mainly relied on resampling or traditional statistical modeling, such as SMOTE and its variants (Lee et al., 2022; Kim et al., 2022), but often struggled to capture complex global joint distributions (Hansen et al., 2023). Later, deep generative models were introduced, including GAN-based methods (such as CTGAN (Xu et al., 2019)), VAE-based methods (such as TVAE (Yadav et al., 2024) and GOGGLE (Liu et al., 2023a)), and autoregressive language-model-based methods (such as GReaT (Borisov et al., 2023)). More recently, diffusion models have become a dominant paradigm for tabular generation (Li et al., 2026). These include methods that directly model data in the original space, such as

TabDDPM (Kotelnikov et al., 2023) and CoDi (Lee et al., 2023), and latent-space approaches, such as TABSYN (Zhang et al., 2024). Despite their strong performance, most of these methods still operate within a unified generative framework over all dimensions (Long et al., 2025; Wang et al., 2024). Recent work has further introduced feature-wise adaptation to better account for tabular heterogeneity. TABDIFF proposes a continuous-time joint diffusion framework for mixed-type tabular data and uses feature-wise learnable noise schedules to adapt to inter-column heterogeneity (Shi et al., 2025). CDTD similarly introduces feature-wise adaptation within a continuous diffusion process (Mueller et al., 2025). These approaches primarily address heterogeneity through the perturbation process or feature-wise parameterization within a shared generative framework. DN-Flow instead focuses on the structure of the learned velocity field: it retains a full-dimensional base transport path while selectively introducing endpoint-guided refinement on numerical dimensions.

DN-Flow builds on conditional flow matching, which directly learns a time-dependent velocity field that transports samples from a simple prior distribution to the data distribution without requiring ODE simulation during training (Lipman et al., 2023). Closely related Rectified Flow formulations learn continuous transport fields along simple interpolation paths and aim to obtain straighter trajectories that can be integrated efficiently (Liu et al., 2023b). Cascaded flow-matching models further demonstrate that complex generation processes can be decomposed into multiple sequential transport stages (Prajwal et al., 2024). DN-Flow differs in that its Driver and Navigator are evaluated jointly at each ODE step rather than sequentially. More generally, the Driver–Navigator formulation is not a generic residual correction or a symmetric expert decomposition: the Navigator first predicts an endpoint and induces a route toward it, while the Driver provides the full-dimensional base transport. The two branches therefore have distinct and non-interchangeable roles, yielding an asymmetric and selectively controlled decomposition of the velocity field.

3 Proposed Method

Let the original tabular dataset be denoted by $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$. We first map each mixed-type row into a unified continuous representation: numerical columns are transformed with a QuantileTransformer, while categorical columns are expanded by one-hot encoding. The resulting sample is denoted by $x_1 \in \mathbb{R}^d$, where $d = d_{\text{num}} + d_{\text{cat}}$, with d_{cat} counting the one-hot-expanded categorical dimensions. DN-Flow learns a continuous-time vector field in this transformed space and maps generated samples back to the original table space after sampling.

3.1 Conditional Flow-Matching Formulation

As illustrated in Figure 1(a), we use the standard linear conditional flow-matching path. Given a real sample x_1 and an initial noise sample $x_0 \sim \mathcal{N}(0, I)$, an intermediate state at time $t \in [0, 1]$ is constructed as

$$x_t = tx_1 + (1-t)x_0. \quad (1)$$

The corresponding target velocity is

$$u^\star(x_t, t) = x_1 - x_0. \quad (2)$$

DN-Flow uses this path and parameterizes the velocity field with the Driver–Navigator structure introduced below.

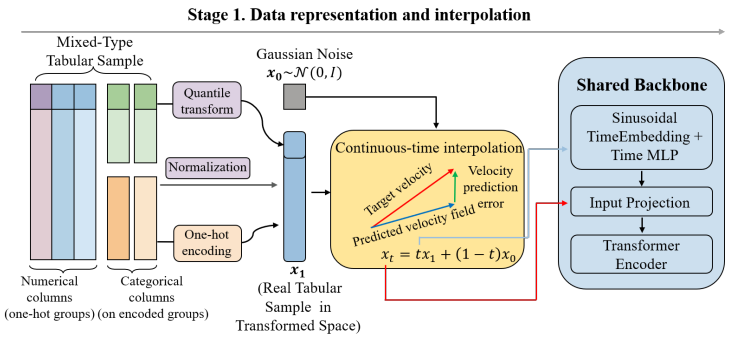


Figure 1(a): Preprocessing and continuous-time interpolation.

3.2 Driver–Navigator Structured Velocity Field Framework

As illustrated in Figure 1(b), given x_t and t, a shared backbone first produces a hidden representation

$$h = \text{Backbone}(x_t, t). \quad (3)$$

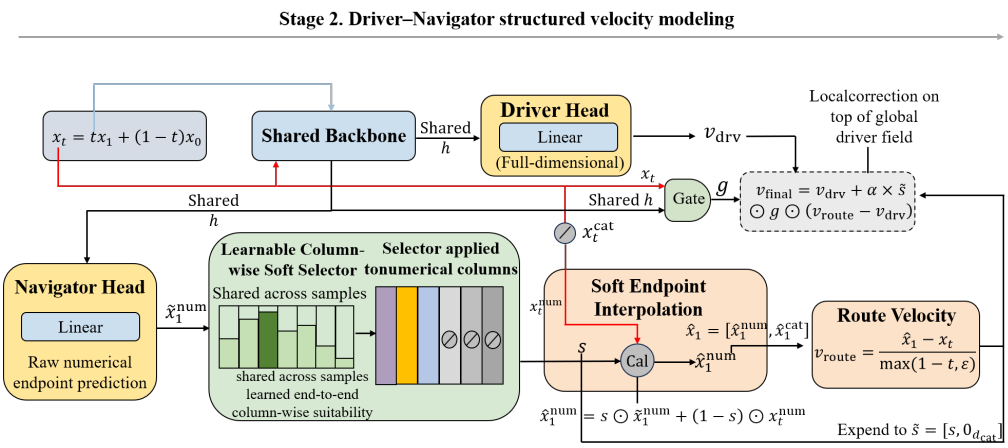


Figure 1(b): Driver–Navigator structured velocity modeling.

The Driver predicts a full-dimensional base velocity

$$v_{\text{drv}} = f_{\text{drv}}(h), \qquad v_{\text{drv}} \in \mathbb{R}^d, \tag{4}$$

while the Navigator predicts a numerical endpoint used to construct a route velocity. A column-wise selector and a dynamic gate control the resulting correction before fusion.

To encode state and time, t is mapped by sinusoidal embeddings followed by a time MLP, yielding $e(t)$. The current state is projected to the same hidden dimension and combined with the time embedding:

$$h_0 = W_{\text{in}} x_t + e(t). \tag{5}$$

The resulting row-level representation is processed by stacked TransformerEncoder blocks as the shared nonlinear backbone:

$$h = \text{TransformerEnc}(h_0). \tag{6}$$

3.3 Driver: Global Base Velocity Field

Given $h \in \mathbb{R}^{d_h}$, the Driver uses a linear readout

$$v_{\text{drv}} = W_{\text{drv}} h + b_{\text{drv}}, \tag{7}$$

where $W_{\text{drv}} \in \mathbb{R}^{d \times d_h}$ and $b_{\text{drv}} \in \mathbb{R}^d$. Although the readout is linear, the full mapping remains nonlinear through the shared backbone. The Driver is also directly supervised by $\mathcal{L}_{\text{drv}}$ in Section 3.5.

3.4 Navigator: Endpoint-Guided Local Route Refinement

3.4.1 Learnable Column-wise Soft Selector

For the numerical dimensions, we introduce a learnable parameter $\beta \in \mathbb{R}^{d_{\text{num}}}$ and define

$$s = \sigma(\beta), \qquad s \in (0,1)^{d_{\text{num}}}. \tag{8}$$

The selector is shared across samples and time steps and learned end-to-end. To apply it in the full transformed space, we use

$$\tilde{s} = [s, 0_{d_{\text{cat}}}] \in [0,1]^d. \tag{9}$$

Thus, the Driver models all dimensions, while Navigator refinement is restricted to numerical dimensions. The continuous-endpoint motivation and categorical counterfactual analysis are provided in Appendix B.3.

3.4.2 Endpoint Prediction

Given h, the Navigator uses a linear head $f_{\text{nav}} : \mathbb{R}^{d_h} \to \mathbb{R}^{d_{\text{num}}}$ to predict a raw numerical endpoint:

$$\tilde{x}_1^{\text{num}} = f_{\text{nav}}(h). \tag{10}$$

The selector interpolates between this prediction and the current numerical state:

$$\hat{x}_1^{\text{num}} = s \odot \widetilde{x}_1^{\text{num}} + (1 - s) \odot x_t^{\text{num}}. \tag{11}$$

For categorical dimensions,

$$\hat{x}_1^{\text{cat}} = x_t^{\text{cat}}, \tag{12}$$

and the complete endpoint is

$$\hat{x}_1 = [\hat{x}_1^{\text{num}}, \hat{x}_1^{\text{cat}}]. \tag{13}$$

3.4.3 Route Velocity and Gated Refinement

Requiring the current state to reach $\hat{x}_1$ at $t = 1$ under a constant route velocity gives

$$v_{\text{route}} = \frac{\hat{x}_1 - x_t}{\max(1 - t, \varepsilon)}, \tag{14}$$

where $\varepsilon > 0$ prevents numerical instability near $t = 1$.

The dynamic gate is defined as

$$g = \sigma(f_{\text{gate}}([h, x_t])), \qquad g \in (0, 1)^d, \tag{15}$$

with

$$\text{Linear}(d_h + d, d_h) \to \text{GELU} \to \text{Linear}(d_h, d) \to \text{Sigmoid}.$$

Here, h provides time-conditioned context and x_t preserves the current per-dimension state.

The final velocity field is

$$v_{\text{final}}(x_t, t) = v_{\text{drv}} + \alpha \widetilde{s} \odot g \odot (v_{\text{route}} - v_{\text{drv}}), \tag{16}$$

where $\alpha > 0$ controls the global correction strength. The selector acts in both endpoint interpolation and route injection. For a numerical dimension j,

$$\begin{aligned} v_{\text{final},j} = &(1 - \alpha s_j g_j) v_{\text{drv},j} \\ &+ \frac{\alpha s_j^2 g_j (\widetilde{x}_{1,j}^{\text{num}} - x_{t,j}^{\text{num}})}{\max(1 - t, \varepsilon)}. \end{aligned} \tag{17}$$

Thus, $s_j \to 0$ recovers the Driver-only field; for categorical dimensions, $\widetilde{s}_j = 0$ and the final velocity is given by the Driver path.

3.5 Training Objective

DN-Flow is trained with

$$\begin{aligned} \mathcal{L} = &\mathcal{L}_{\text{flow}} + \lambda_{\text{drv}} \mathcal{L}_{\text{drv}} + \lambda_{x_1} \mathcal{L}_{x_1} + \lambda_{\text{corr}} \mathcal{L}_{\text{corr}} \\ &+ \lambda_{\text{sp}} \mathcal{L}_{\text{sparse}} + \lambda_{\text{bin}} \mathcal{L}_{\text{binary}}. \end{aligned} \tag{18}$$

The main flow loss supervises the fused velocity,

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\Big[\|v_{\text{final}}(x_t, t) - (x_1 - x_0)\|_2^2\Big], \tag{19}$$

and the Driver loss directly supervises the base field,

$$\mathcal{L}_{\text{drv}} = \mathbb{E}\Big[\|v_{\text{drv}} - (x_1 - x_0)\|_2^2\Big]. \tag{20}$$

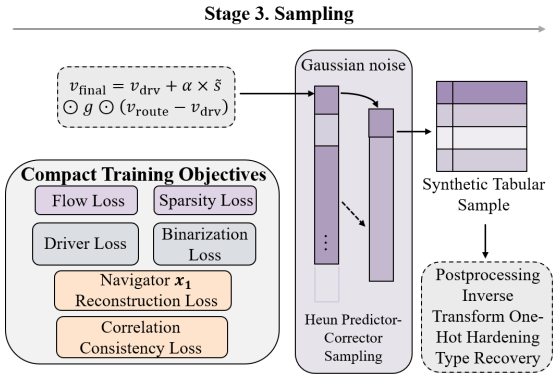


Figure 1(c): Training objectives and sampling.

$\mathcal{L}_{x_1}$ supervises numerical endpoint reconstruction and $\mathcal{L}_{\text{corr}}$ preserves numerical correlation structure. The selector regularizers are

$$\mathcal{L}_{\text{sparse}} = \frac{1}{d_{\text{num}}} \sum_j s_j, \qquad \mathcal{L}_{\text{binary}} = \frac{1}{d_{\text{num}}} \sum_j s_j(1 - s_j), \tag{21}$$

which discourage all-active and persistently ambiguous selector values, respectively. Full definitions and derivations are given in Appendix B.4. Loss weights are selected using validation data; the search procedure and dataset-specific settings are reported in Appendix D.

3.6 Training and Sampling Overview

As illustrated in Figure 1(c), during training, we sample $x_1 \sim p_{\mathrm{data}}$, $x_0 \sim \mathcal{N}(0, I)$, and $t \sim \mathcal{U}(0, 1)$, construct $x_t = tx_1 + (1-t)x_0$, and jointly optimize the shared backbone, Driver, Navigator, selector, and gate using the objective defined above.

During sampling, generation starts from Gaussian noise and integrates v_{final} from $t = 0$ to $t = 1$ using a second-order Heun predictor–corrector solver. The resulting continuous representation is then mapped back to the original tabular space through inverse numerical transformation, categorical one-hot hardening, and type recovery. Detailed training and sampling pseudocode is provided in Appendix C.

4 Experiments

To evaluate DN-Flow, we conducted experiments on multiple real-world tabular datasets, including benchmark datasets from the official codebase of (Shi et al., 2025) and two additional public datasets, introduced to assess generalization beyond the original benchmark. For the benchmark datasets, we directly reused the official dataset files, data splits, and configuration files from (Shi et al., 2025) without modification. For the additional datasets, we followed the same experimental pipeline and retrained the major baselines under the same evaluation protocol. We evaluate generation quality in terms of data fidelity, downstream utility, and supplementary privacy and robustness analyses.

4.1 Experimental Setup

Datasets. We evaluated DN-Flow on eight real-world mixed-type tabular datasets: Adult, Default, Shoppers, Magic, Beijing, News, Bank Marketing, and Bike Sharing. In the tables, Bank Marketing and Bike Sharing are abbreviated as BM and BS, respectively. Adult, Default, Shoppers, Magic, and BM are classification datasets, whereas Beijing, News, and BS are regression datasets. Detailed dataset statistics are provided in Appendix A.1.

Baselines. We compared DN-Flow with eleven representative tabular data generation methods covering several major paradigms: the traditional resampling method SMOTE; the GAN-based method CTGAN; the VAE-based methods TVAE and GOGGLE; the autoregressive language-model-based method GReaT; and the diffusion-based methods STaSy, CoDi, CDTD, TabDDPM, TABSYN, and TABDIFF.

Evaluation Metrics. Following the evaluation protocol of (Shi et al., 2025), we assessed synthetic data quality from three perspectives. (1) *Data Fidelity*, including Shape and Trend, where Shape measures the recovery of column-wise marginal distributions and Trend evaluates pairwise inter-column dependencies. We also report supplementary fidelity metrics such as α-Precision, β-Recall, and C2ST. (2) *Downstream Task Utility*, evaluated primarily by Machine Learning Efficiency (MLE) and missing-value imputation. (3) *Privacy*, primarily evaluated by Distance to Closest Records (DCR) (Zheng & Charoenphakdee, 2022). Detailed definitions are provided in Appendix A.2.

Repeated Runs and Statistical Reporting. Each model was trained with five independent training seeds, and 20 sampling seeds were used for the best validation checkpoint of each run. We first averaged results over the 20 sampling seeds within each run and then reported mean $\pm$ std across the five training seeds, treating the training seed as the unit of statistical reporting. Unless otherwise specified, all results follow this protocol.

Hyperparameter Selection. The same candidate grids and fixed-order coordinate-wise search were used across all datasets using only training and validation data. Candidates were first evaluated with one training seed and three sampling seeds, and the strongest were re-evaluated with three training seeds and five sampling seeds; selection used normalized validation Shape and Trend errors. The test set was used only for final evaluation, while baselines followed their official or recommended configurations; full details are provided in Appendix D.

Table 1: Performance comparison on the error rates (%) of Shape. Lower is better.

Method	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
SMOTE	1.60±0.23	1.48±0.15	2.68±0.19	0.91±0.05	1.85±0.21	5.31±0.46	2.74±0.21	4.33±0.21	2.61
CTGAN	16.84±0.03	16.83±0.04	21.15±0.10	9.81±0.08	21.39±0.05	16.09±0.02	15.33±0.06	22.87±0.07	17.54
TVAE	14.22±0.08	10.17±0.05	24.51±0.06	8.25±0.06	19.16±0.06	16.62±0.03	13.99±0.06	21.84±0.07	16.09
GOGGLE	16.97	17.02	22.33	1.90	16.93	25.32	18.25	25.55	18.03
GReaT	12.12±0.04	19.94±0.06	14.51±0.12	16.16±0.09	8.25±0.12	OOM	16.44±0.09	22.74±0.10	16.56
STaSy	11.29±0.06	5.77±0.06	9.37±0.09	6.29±0.13	6.71±0.03	6.89±0.03	6.26±0.07	10.23±0.08	7.85
CoDi	21.38±0.06	15.77±0.07	31.84±0.05	11.56±0.26	16.94±0.02	32.27±0.04	22.39±0.08	32.96±0.09	23.14
CDTD	0.97±0.04	0.86±0.08	1.10±0.08	0.73±0.09	0.92±0.04	2.18±0.03	1.12±0.07	1.79±0.08	1.21
TabDDPM	1.75±0.03	1.57±0.08	2.72±0.13	1.01±0.09	1.30±0.03	78.75±0.01	32.08±0.12	42.42±0.14	20.20
TABSYN	0.81±0.05	1.01±0.08	1.44±0.07	1.03±0.14	1.26±0.05	2.06±0.04	1.24±0.06	2.90±0.08	1.47
TABDIFF	0.63±0.05	1.24±0.07	1.28±0.09	0.78±0.08	1.03±0.05	2.35±0.03	1.15±0.03	1.65±0.07	1.26
DN-Flow	**0.38±0.03**	**0.20±0.02**	**0.94±0.07**	**0.59±0.06**	**0.55±0.03**	**1.98±0.02**	**0.73±0.05**	**1.42±0.07**	**0.85**

Note: For GOGGLE, we followed the official implementation with a fixed random seed during sampling. Hence, its sample-based fidelity results are reported as single values, while MLE is reported with variance under repeated evaluation.

Table 2: Performance comparison on the error rates (%) of Trend. Lower is better.

Method	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
SMOTE	3.28±0.29	8.41±0.38	3.56±0.22	3.16±0.41	2.39±0.35	5.38±0.76	4.85±0.45	9.12±0.68	5.02
CTGAN	20.23±1.20	26.95±0.93	13.08±0.16	7.00±0.19	22.95±0.08	5.37±0.05	21.05±0.62	29.40±1.15	18.25
TVAE	14.15±0.88	19.50±0.95	18.67±0.38	5.82±0.49	18.01±0.08	6.17±0.09	14.88±0.35	28.91±0.72	15.76
GOGGLE	45.29	21.94	23.90	9.47	45.94	23.19	35.12	52.44	32.16
GReaT	17.59±0.22	70.02±0.12	45.16±0.18	10.23±0.40	59.60±0.55	OOM	45.33±0.41	78.65±0.88	46.65
STaSy	14.51±0.25	5.96±0.26	8.49±0.15	6.61±0.53	8.00±0.10	3.07±0.04	10.15±0.18	14.28±0.31	8.88
CoDi	22.49±0.08	68.41±0.05	17.78±0.11	6.53±0.25	7.07±0.15	11.10±0.01	24.50±0.15	47.32±0.25	25.65
CDTD	2.24±0.14	1.86±0.33	1.92±0.10	0.78±0.16	1.95±0.12	**1.12±0.04**	1.88±0.10	3.55±0.09	1.91
TabDDPM	3.01±0.25	4.89±0.10	6.61±0.16	1.70±0.22	2.71±0.09	13.16±0.11	45.25±0.15	69.68±0.18	18.38
TABSYN	1.93±0.07	2.81±0.48	2.13±0.10	0.88±0.18	3.13±0.34	1.52±0.03	1.95±0.09	4.70±0.09	2.38
TABDIFF	1.49±0.16	2.55±0.75	**1.74±0.08**	**0.76±0.12**	2.59±0.15	1.28±0.04	2.31±0.10	4.10±0.09	2.10
DN-Flow	**1.15±0.12**	**1.00±0.17**	2.10±0.10	0.84±0.13	**1.61±0.09**	1.16±0.04	**1.60±0.07**	**2.95±0.06**	**1.55**

4.2 Data Fidelity Analysis

Shape and Trend. We first evaluated synthetic-data fidelity using Shape and Trend, where lower values indicate better performance. Tables 1 and 2 report the results on all eight datasets. On Shape, DN-Flow achieves the lowest error on all eight datasets, with an average error of 0.85, compared with 1.21 for CDTD, 1.26 for TABDIFF, and 1.47 for TABSYN. This corresponds to reductions of 29.8%, 32.5%, and 42.2% in average Shape error, respectively.

For Trend, DN-Flow obtains the lowest average error of 1.55. It ranks first on Adult, Default, Beijing, BM, and BS, second on News, and third on Shoppers and Magic. CDTD performs better on Shoppers, Magic, and News, while DN-Flow achieves a lower average Trend error than CDTD (1.55 vs. 1.91). These results indicate a strong and consistent numerical advantage in marginal fidelity and competitive dependency preservation, while we do not claim statistically decisive superiority on every individual dataset.

Type-wise Fidelity. Because the Navigator refinement operates only on numerical dimensions, we additionally decomposed the existing Shape and Trend evaluations by variable type. Numerical Shape and numerical–numerical dependency errors improve from Base to DN-Flow on all eight datasets, while categorical Shape also improves on all eight datasets. Some categorical–categorical and numerical–categorical pairwise terms do not improve uniformly, indicating that the gains are not confined to numerical marginals but that the strongest direct effect of Navigator remains on continuous dimensions. The complete type-wise decomposition is reported in Appendix E.

Additional Fidelity Metrics and Privacy. Beyond Shape and Trend, we further evaluated the generated data using α-Precision, β-Recall, and C2ST, together with the DCR privacy diagnostic. Owing to space limitations, these supplementary results are presented in Appendix E.

4.3 Downstream Task Performance

Machine Learning Efficiency. To evaluate downstream utility, we adopted Machine Learning Efficiency (MLE). Each generative model was trained on the real training set and used to generate a synthetic training set of the same size, after which an XGBoost classifier or regressor was trained on the synthetic data and evaluated on the real test set. AUC was used for classification tasks and RMSE for regression tasks. To aggregate the two metrics on different scales, we first compute the relative

Table 3: Evaluation of Machine Learning Efficiency (MLE), where AUC and RMSE are used for classification and regression tasks, respectively. Average Gap is the mean normalized relative deviation from the real-data reference across datasets.

Methods	Adult (AUC↑)	Default (AUC↑)	Shoppers (AUC↑)	Magic (AUC↑)	Beijing (RMSE↓)	News (RMSE↓)	BM (AUC↑)	BS (RMSE↓)	Average Gap (%)
Real	0.927±.000	0.770±.005	0.926±.001	0.946±.001	0.423±.003	0.842±.002	0.935±.001	0.385±.002	0.00%
CTGAN	0.886±.002	0.696±.005	0.875±.009	0.855±.006	0.902±.019	0.880±.016	0.871±.008	0.785±.021	32.21%
TVAE	0.878±.004	0.724±.005	0.871±.006	0.887±.003	0.770±.011	1.01±.016	0.865±.005	0.820±.015	30.74%
GOGGLE	0.778±.012	0.584±.005	0.658±.052	0.654±.024	1.09±.025	0.877±.002	0.695±.018	1.15±.030	60.78%
GReaT	0.913±.003	0.755±.006	0.902±.005	0.888±.008	0.653±.013	OOM	0.898±.005	0.612±.014	18.50%
STaSy	0.906±.001	0.752±.006	0.914±.005	0.934±.003	0.656±.014	0.871±.002	0.905±.004	0.635±.011	16.73%
CoDi	0.871±.006	0.525±.006	0.865±.006	0.932±.003	0.818±.021	1.21±.005	0.862±.007	0.985±.025	43.33%
CDTD	0.909±.002	0.765±.003	0.917±.004	0.936±.003	0.565±.012	0.864±.019	0.918±.003	0.442±.020	7.18%
TabDDPM	0.907±.001	0.758±.004	0.918±.005	0.935±.003	0.592±.011	4.86±3.04	0.658±.002	3.05±0.21	155.59%
TABSYN	0.909±.001	0.763±.002	0.914±.004	0.937±.002	0.580±.009	0.862±.024	0.915±.004	0.488±.008	9.19%
TABDIFF	0.912±.002	0.763±.005	0.921±.004	0.936±.003	0.555±.013	0.866±.021	0.910±.003	0.531±.12	9.85%
DN-Flow	**0.913±.002**	**0.771±.005**	**0.925±.004**	**0.938±.004**	**0.553±.013**	**0.857±.018**	**0.923±.003**	**0.410±.006**	**5.37%**

Table 4: Comparison of Shape results for different ablation variants across datasets. Lower is better.

Model	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
Base	0.386±0.031	0.234±0.023	0.987±0.074	0.623±0.063	0.574±0.031	2.121±0.021	0.79±0.06	1.49±0.08	0.901
w/o X1Recon	0.445±0.035	0.211±0.021	0.987±0.074	0.595±0.061	0.556±0.030	1.997±0.020	0.77±0.05	1.46±0.07	0.878
w/o DriverLoss	0.419±0.033	0.203±0.020	1.484±0.111	0.599±0.061	0.577±0.031	2.047±0.021	0.83±0.08	1.62±0.11	0.972
w/o Selector	0.478±0.038	0.207±0.021	1.139±0.085	0.590±0.060	0.581±0.032	2.000±0.020	0.80±0.06	1.55±0.09	0.918
w/o Gate	0.381±0.031	0.201±0.021	1.459±0.109	0.696±0.071	0.946±0.052	2.071±0.021	0.92±0.09	1.84±0.12	1.064
w/o Decomposition	0.420±0.033	0.215±0.022	1.085±0.081	0.615±0.062	0.585±0.032	2.035±0.020	0.78±0.05	1.52±0.08	0.907
DN-Flow	**0.380±0.030**	**0.200±0.020**	**0.940±0.070**	**0.590±0.060**	**0.550±0.030**	**1.980±0.020**	**0.73±0.05**	**1.42±0.07**	**0.849**

Table 5: Comparison of Trend results for different ablation variants across datasets. Lower is better.

Model	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
Base	1.183±0.123	1.007±0.171	2.046±0.097	0.841±0.131	1.759±0.098	1.224±0.042	1.65±0.08	3.07±0.07	1.598
w/o X1Recon	1.244±0.130	1.021±0.174	2.249±0.107	0.882±0.137	1.655±0.093	1.168±0.040	1.68±0.08	3.13±0.07	1.629
w/o DriverLoss	1.242±0.130	1.006±0.169	3.532±0.168	0.851±0.132	1.632±0.091	1.194±0.041	1.70±0.09	3.19±0.08	1.793
w/o Selector	1.352±0.141	1.004±0.170	2.644±0.126	0.840±0.130	1.700±0.095	1.162±0.040	1.73±0.09	3.10±0.07	1.692
w/o Gate	1.175±0.123	1.261±0.214	3.366±0.160	0.994±0.154	2.601±0.145	1.190±0.041	2.00±0.15	3.84±0.14	2.053
w/o Decomposition	1.214±0.127	1.015±0.173	2.345±0.112	0.865±0.134	1.685±0.094	1.175±0.041	1.71±0.08	3.22±0.08	1.654
DN-Flow	**1.150±0.120**	**1.000±0.170**	2.100±0.100	**0.840±0.130**	**1.610±0.090**	**1.160±0.040**	**1.60±0.07**	**2.95±0.06**	**1.551**

gap to the real-data score within each dataset, $\mathrm{Gap}_d = |m_d^{\mathrm{syn}} - m_d^{\mathrm{real}}|/|m_d^{\mathrm{real}}|$, and then average it across the eight datasets.

Table 3 reports the MLE results. DN-Flow achieves the lowest Average Gap of 5.37%, compared with 7.18% for CDTD, 9.19% for TABSYN, and 9.85% for TABDIFF. These results indicate that DN-Flow preserves strong predictive utility for downstream tasks. We treat MLE as a complementary measure of downstream utility rather than a standalone indicator of overall generation quality.

4.4 Ablation Studies

To assess the contribution of the main components in DN-Flow, we considered six variants in addition to the full model. Base retains the shared time embedding, input projection, nonlinear backbone, and a single full-dimensional velocity head, without the Navigator correction path. The other variants retain the correction path but remove one of its supporting mechanisms: w/o X1Recon removes endpoint reconstruction supervision; w/o DriverLoss removes direct supervision on the Driver; w/o Selector sets all numerical selector values to one; w/o Gate removes sample- and time-dependent gating; and w/o Decomposition replaces the role-specialized Driver–Navigator field with a parameter-matched single-branch predictor.

Compared with Base, full DN-Flow reduces average Shape from 0.901 to 0.849 and average Trend from 1.598 to 1.551, corresponding to improvements of 5.8% and 2.9%, respectively. Shape improves on all eight datasets and Trend improves on seven of eight. These gains are modest on several individual datasets and should not be interpreted as statistically decisive improvements in every case.

The ablations are not a nested sequence of increasingly complete models. Most variants retain the local correction path while removing one mechanism that regulates or supervises it. In particular, w/o Gate produces the largest degradation, illustrating the importance of controlling correction strength as a function of sample and time. Removing the Selector also degrades both averages, while w/o DriverLoss shows that direct supervision of the base field is important for reliable transport. The parameter-matched w/o Decomposition variant performs slightly worse than Base on average,

Table 6: Parameter-matched backbone comparison averaged over eight datasets.

Backbone	Parameters	Shape ↓	Trend ↓
Matched MLP	13,665,502	0.868	1.607
Transformer blocks	13,677,790	**0.849**	**1.551**

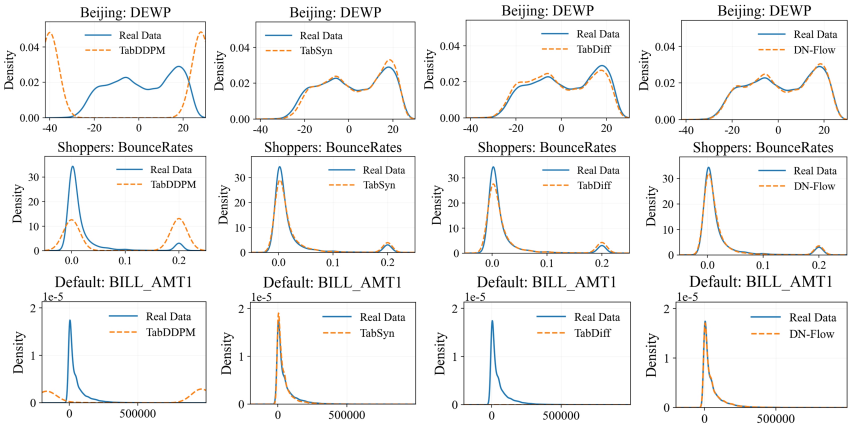


Figure 2: Comparison of the marginal distributions of representative numerical columns between real data and data generated by different methods.

indicating that simply adding comparable single-branch capacity does not reproduce the gain of the role-specialized decomposition.

Backbone Robustness. To test whether the method depends on the TransformerEncoder-block backbone, we replaced only the shared backbone with a parameter-matched residual MLP while keeping preprocessing, prediction heads, losses, training seeds, and sampling settings unchanged.

The parameter difference is only 0.09%. The Transformer-block implementation gives a modest numerical advantage, with 2.2% lower average Shape and 3.5% lower average Trend. A paired analysis over matched training seeds favors the Transformer implementation in all 16 dataset–metric comparisons, but the 95% confidence interval excludes zero in only 4 of the 16 comparisons. We therefore interpret this result as evidence that DN-Flow is not tied to a specific backbone, rather than as evidence that the Transformer backbone is uniformly superior.

Selector Stability. We further examined the learned column-wise selector across the five training seeds. Selector rankings are positively correlated across seeds on all eight datasets, with a macro-average rank correlation of 0.79. The correlation between selector values and column-wise Shape improvement is also positive on all eight datasets, averaging 0.60, and DN-Flow improves 91 of 104 numerical columns relative to Base. These results support the selector as a stable column-level refinement control parameter, although it should not be interpreted as a conventional feature-importance score because the final correction also depends on the dynamic gate, endpoint quality, and the discrepancy between Driver and route velocities. Detailed results are provided in Appendix E.

Additional analyses of the numerical-only Navigator, timestep-dependent gate behavior, statistical reliability, solver/NFE trade-offs, and hyperparameter sensitivity are provided in Appendices E and F.

4.5 Visualizations of Synthetic Data

To complement the quantitative Shape and Trend results, we further visualize synthetic-data fidelity from three perspectives. For numerical features (Figure 2), DN-Flow closely matches the real marginal distributions on representative columns such as Beijing: DEWP, Shoppers: BounceRates, and Default: BILL_AMT1.

For categorical features (Figure 3), DN-Flow also produces distributions that are generally close to the real data, although it is not uniformly the best method on every individual column. This is consistent with the type-wise analysis above: the Driver models the full mixed-type table, while the specialized Navigator correction acts directly only on numerical dimensions.

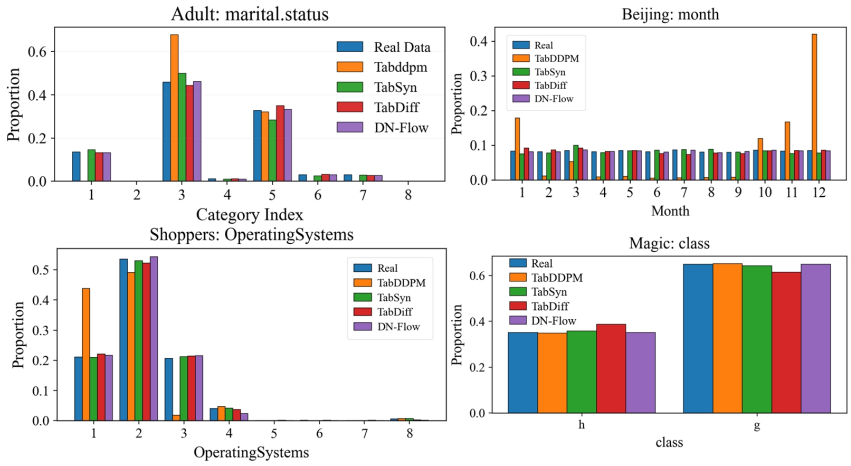


Figure 3: Comparison of the frequency distributions of representative categorical columns between real data and data generated by different methods.

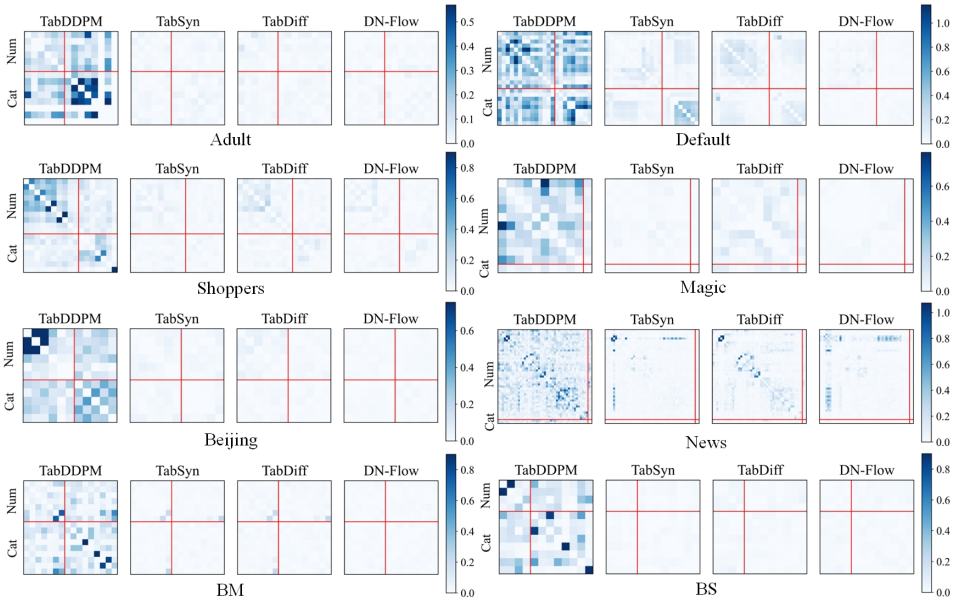


Figure 4: Comparison of pairwise correlation error heatmaps between real data and data generated by representative strong baselines. Darker colors indicate larger errors, whereas lighter colors indicate smaller errors.

Figure 4 provides a qualitative comparison of pairwise dependency errors against representative strong baselines. DN-Flow generally produces lighter error patterns on several datasets, indicating better preservation of pairwise dependencies in these cases.

5 Conclusion

In this paper, we proposed DN-Flow, a structured flow-matching framework for mixed-type tabular data generation. DN-Flow departs from homogeneous velocity modeling by explicitly separating globally stable transport from controlled local route-aware refinement through an asymmetric Driver–Navigator decomposition. To model tabular heterogeneity more explicitly, it further introduces a column-wise soft selector and a dynamic gate, which together characterize column-level suitability for correction and instance-/time-dependent correction strength. Experiments on eight real-world datasets show that DN-Flow achieves strong overall performance in fidelity, downstream utility, and supplementary metrics, while ablation studies confirm the importance of its structural decomposition and controlled local correction mechanism.

Acknowledgments and Disclosures

Funding. This work was supported by the National Key Research and Development Program of China (Grant No. 2024YFB3311901) and the National Natural Science Foundation of China (Grant No. 62372154).

Competing Interests. The authors declare no competing interests.

References

Alaa, A., Van Breugel, B., Saveliev, E. S., and van der Schaar, M. How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 290–306, 17–23 Jul 2022.

Borisov, V., Sessler, K., Leemann, T., Pawelczyk, M., and Kasneci, G. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations*, 2023.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, pp. 507–520, 2022.

Hansen, L., Seedat, N., van der Schaar, M., and Petrovic, A. Reimagining synthetic tabular data generation through data-centric ai: A comprehensive benchmark. In *Advances in Neural Information Processing Systems*, volume 36, pp. 33781–33823, 2023.

Holzmüller, D., Grinsztajn, L., and Steinwart, I. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 26577–26658, 2024.

Kim, J., Lee, C., Shin, Y., Park, S., Kim, M., Park, N., and Cho, J. Sos: Score-based oversampling for tabular data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 762–772, 2022.

Kim, J., Lee, C., and Park, N. STasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2023.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. 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, 23–29 Jul 2023.

Lee, C., Kim, J., and Park, N. CoDi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18940–18956, 23–29 Jul 2023.

Lee, J., Kim, M., Jeong, Y., and Ro, Y. Differentially private normalizing flows for synthetic tabular data generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7345–7353, Jun. 2022.

Li, Z., Gan, J., and Qi, J. Towards synthesizing high-dimensional tabular data with limited samples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(18):15207–15215, Mar. 2026.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., Yin, H.-H., Zhou, T., Jiang, J.-P., and Ye, H.-J. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226):1–16, 2025.

Liu, T., Qian, Z., Berrevoets, J., and van der Schaar, M. GOGGLE: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2023a.

Liu, X., Gong, C., and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b.

Long, Y., Xu, L., and Brintrup, A. Evaluating inter-column logical relationships in synthetic tabular data generation. In *Will Synthetic Data Finally Solve the Data Access Problem?*, 2025.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. When do neural nets outperform boosted trees on tabular data? In *Advances in Neural Information Processing Systems*, volume 36, pp. 76336–76369, 2023.

Mueller, M., Gruber, K., and Fok, D. Continuous diffusion for mixed-type tabular data. In *International Conference on Learning Representations*, volume 2025, pp. 39186–39225, 2025.

Prajwal, K. R., Shi, B., Le, M., Vyas, A., Tjandra, A., Luthra, M., Guo, B., Wang, H., Afouras, T., Kant, D., and Hsu, W.-N. MusicFlow: Cascaded flow matching for text guided music generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41052–41063, 21–27 Jul 2024.

Qian, Z., Davis, R., and van der Schaar, M. Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3173–3188, 2023.

Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., and Leskovec, J. Tabdiff: a mixed-type diffusion model for tabular data generation. In *International Conference on Learning Representations*, volume 2025, pp. 37353–37375, 2025.

Van Breugel, B. and Van Der Schaar, M. Position: Why tabular foundation models should be a research priority. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48976–48993, 21–27 Jul 2024.

Wang, A. X., Chukova, S. S., Simpson, C. R., and Nguyen, B. P. Challenges and opportunities of generative models on tabular data. *Applied Soft Computing*, 166:112223, 2024.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yadav, P., Gaur, M., Madhukar, R. K., Verma, G., and Kumar, P. Rigorous experimental analysis of tabular data generated using tvae and ctgan. *International Journal of Advanced Computer Science & Applications*, 15(4), 2024.

Yan, J., Chen, J., Wu, Y., Chen, D. Z., and Wu, J. T2g-former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):10720–10728, Jun. 2023.

Zhang, H., Zhang, J., Shen, Z., Srinivasan, B., Qin, X., Faloutsos, C., Rangwala, H., and Karypis, G. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *International Conference on Learning Representations*, volume 2024, pp. 52829–52857, 2024.

Zheng, S. and Charoenphakdee, N. Diffusion models for missing value imputation in tabular data. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

A Detailed Experimental Setup

A.1 Datasets

We evaluate DN-Flow on eight widely used real-world tabular datasets[2]: Adult, Default, Shoppers, Magic, Beijing, News, Bank Marketing, and Bike Sharing. All of these datasets contain both numerical and categorical features, and therefore constitute typical mixed-type tabular data. In addition, each dataset is associated with an inherent downstream machine learning task: Adult, Default, Shoppers, Magic, and Bank Marketing are used for classification, while Beijing, News, and Bike Sharing are used for regression. This dataset selection covers a range of data scales, feature compositions, and task types, and thus enables a relatively comprehensive evaluation of the adaptability and stability of DN-Flow across different mixed-type tabular scenarios. Our dataset setup is consistent with the benchmark protocols commonly adopted in existing tabular data generation studies.

Following the benchmark setting, each dataset is divided into training, validation, and test sets. The training set is used to train the generative model, the validation set is used for model selection, and the test set is used for final downstream evaluation. Table 7 summarizes the basic statistics of the eight datasets used in this work, including the total number of samples, the number of numerical columns, the number of categorical columns, the maximum categorical cardinality, and the sizes of the training, validation, and test splits. For the benchmark datasets adopted from (Shi et al., 2025), we follow their dataset statistics and split protocol. For Bank Marketing and Bike Sharing, we follow the same preprocessing and split protocol for consistency.

Table 7: Statistics of the datasets used in this work. Num denotes the number of numerical columns, Cat denotes the number of categorical columns, and Max Cat denotes the number of categories of the categorical feature with the largest cardinality.

Dataset	Rows	Num	Cat	Max Cat	Train	Validation	Test	Task
Adult	48,842	6	9	42	28,943	3,618	16,281	Classification
Default	30,000	14	11	11	24,000	3,000	3,000	Classification
Shoppers	12,330	10	8	20	9,864	1,233	1,233	Classification
Magic	19,019	10	1	2	15,215	1,902	1,902	Classification
Beijing	43,824	7	5	31	35,058	4,383	4,383	Regression
News	39,644	46	2	7	31,714	3,965	3,965	Regression
BM	45,211	7	9	12	36,620	4,069	4,522	Classification
BS	17,389	4	8	24	14,085	1,565	1,739	Regression

A.2 Evaluation Metrics

To more comprehensively evaluate the quality of synthetic tabular data, we assess the model from three perspectives: data fidelity, downstream utility, and privacy protection. In the main paper, we primarily report three core metrics, namely Shape, Trend, and Machine Learning Efficiency (MLE), while the remaining supplementary metrics are presented in the appendix for a more complete analysis of model performance. This evaluation protocol is consistent with commonly adopted benchmark settings in recent studies on tabular data generation.

A.2.1 Shape and Trend

We use Shape and Trend to evaluate the fidelity of synthetic data[3]. Both metrics are adopted from SDMetrics. Specifically, Shape measures how well the synthetic data recover the marginal distribution of each individual column, while Trend measures how well the synthetic data preserve pairwise dependencies between columns. For both metrics, lower error values indicate that the synthetic data are closer to the real data.

[2] `https://archive.ics.uci.edu/datasets`

[3] `https://docs.sdv.dev/sdmetrics`

(1) Shape. For numerical columns, Shape uses the Kolmogorov–Smirnov Test (KST) to measure the discrepancy between the distributions of real and synthetic data. Let $F_r(x)$ and $F_s(x)$ denote the cumulative distribution functions of the real and synthetic data, respectively. Then KST is defined as the maximum difference between the two:

$$\mathrm{KST}(r, s) = \sup_x |F_r(x) - F_s(x)| \tag{22}$$

where the cumulative distribution function $F(x)$ is defined as

$$F(x) = \int_{-\infty}^{x} p(x)\, dx \tag{23}$$

For categorical columns, Shape uses Total Variation Distance (TVD) to measure the discrepancy between category frequency distributions. Let Ω denote the set of all possible values of the categorical column. Then TVD is defined as

$$\mathrm{TVD}(r, s) = \frac{1}{2} \sum_{\omega \in \Omega} |R(\omega) - S(\omega)| \tag{24}$$

where $R(\omega)$ and $S(\omega)$ denote the empirical frequencies of category ω in the real and synthetic data, respectively. In practice, we compute the corresponding error for each column and then average over all columns to obtain the final Shape score. Intuitively, Shape focuses on how well the model recovers the statistical form of each individual feature.

(2) Trend. Trend measures how well the synthetic data recover pairwise dependencies between columns. For continuous column pairs, we first compute the Pearson correlation coefficient:

$$\rho_{x,y} = \frac{\mathrm{Cov}(x, y)}{\sigma_x \sigma_y} \tag{25}$$

where $\mathrm{Cov}(x, y)$ denotes the covariance between columns x and y, and σ_x and σ_y denote their standard deviations, respectively. We then quantify the correlation recovery error for continuous column pairs by comparing the Pearson correlations in the real and synthetic data:

$$\text{Pearson Score} = \frac{1}{2}\, \mathbb{E}_{x,y} \left| \rho^R(x, y) - \rho^S(x, y) \right| \tag{26}$$

where $\rho^R(x, y)$ and $\rho^S(x, y)$ denote the Pearson correlation coefficients of column pair x, y in the real and synthetic data, respectively. Since $\rho \in [-1, 1]$, dividing by 2 normalizes the error into the range $[0, 1]$. For categorical column pairs, Trend uses contingency similarity to measure the discrepancy between contingency distributions in the real and synthetic data. Let A and B be two categorical columns, and let $\alpha \in A$ and $\beta \in B$ denote possible categories in the two columns. The corresponding error is defined as

$$\text{Contingency Score} = \frac{1}{2} \sum_{\alpha \in A} \sum_{\beta \in B} |R_{\alpha,\beta} - S_{\alpha,\beta}| \tag{27}$$

where $R_{\alpha,\beta}$ and $S_{\alpha,\beta}$ denote the joint frequencies of category combination (α, β) in the real and synthetic data, respectively. Finally, we average the correlation errors over all column pairs to obtain the overall Trend score. Unlike Shape, which mainly focuses on single-column distributions, Trend emphasizes whether the model can preserve the relational structure among columns in real tables.

A.2.2 Machine Learning Efficiency (MLE)

To evaluate the practical usefulness of synthetic data in downstream tasks, we adopt Machine Learning Efficiency (MLE) as an evaluation metric. The goal of this metric is not to directly measure distributional fidelity, but rather to assess whether the generated samples preserve the key information needed to support downstream predictive tasks. Therefore, MLE is better understood as a task-oriented utility metric, rather than as a standalone criterion for generation quality. Specifically, for each dataset, we first train the generative model on the real training set, and then sample synthetic data of the same size as the real training set from the trained model. Next, we train a downstream predictor on the synthetic data and evaluate it on the real test set. For classification tasks, we report AUC; for regression tasks, we report RMSE. In addition, we compare the predictor trained on synthetic data with the predictor trained on real data, and quantify the loss in downstream utility by computing the Average Gap across datasets. A smaller gap indicates that the synthetic data better preserve the predictive structure of the real data. This evaluation protocol is consistent with common practice in existing work on tabular data generation.

A.2.3 Distance to Closest Records (DCR)

We also adopt Distance to Closest Records (DCR) as a privacy-oriented evaluation metric. The key idea is that high-quality synthetic data should reflect the overall data manifold, rather than merely memorizing or copying training samples. DCR measures whether synthetic samples are excessively close to the training data by comparing, for each synthetic sample, its nearest-neighbor relationship to the training set and to the holdout/test set. In general, a DCR score closer to 50% indicates that the synthetic samples are not disproportionately concentrated around the training data, and therefore suggests better privacy protection.

A.2.4 Additional Fidelity Metrics

In addition to Shape and Trend (Following (Liu et al., 2023a) and (Alaa et al., 2022)), we further include several supplementary fidelity metrics in the appendix, including α-Precision, β-Recall, and Detection/C2ST. These metrics provide complementary perspectives on synthetic data quality. Specifically, α-Precision measures whether synthetic samples more accurately fall within the support of the real data distribution, β-Recall measures the extent to which the synthetic data cover the support of the real data distribution, and Detection/C2ST evaluates how easily a classifier can distinguish real samples from synthetic ones. In general, higher α-Precision and β-Recall, together with lower distinguishability, indicate better fidelity of the synthetic data. The detailed experimental results for these supplementary metrics are all provided in Appendix E.

B Additional Method Details

B.1 Driver–Navigator Structured Velocity Field

In this section, we provide further clarification of the key equations of the Driver–Navigator structured velocity field introduced in Section 3 of the main text. First, the most important property of Eq. (16) is that it does not simply add the Driver and Navigator branches directly. Instead, it uses the extended selection vector $\widetilde{s}$ and the gating vector g to regulate local refinement on a dimension-wise basis. Expanding Eq. (16) along the j-th dimension gives

$$v_{\mathrm{final},j}(x_t, t) = v_{\mathrm{drv},j} + \alpha\, \widetilde{s}_j g_j \left(v_{\mathrm{route},j} - v_{\mathrm{drv},j}\right) \tag{28}$$

In the current model, $\widetilde{s}_j$ is a continuous weight. This leads to the following interpretations:

1. When $\widetilde{s}_j \approx 0$, we have:
$$v_{\mathrm{final},j}(x_t, t) \approx v_{\mathrm{drv},j} \tag{29}$$
which means that the update on this dimension almost completely degenerates to the Driver-only form, and the effect of Navigator refinement is very weak.
2. When $\widetilde{s}_j \approx 1$, we have:
$$v_{\mathrm{final},j}(x_t, t) \approx v_{\mathrm{drv},j} + \alpha g_j \left(v_{\mathrm{route},j} - v_{\mathrm{drv},j}\right) \tag{30}$$
which means that this dimension relies more strongly on the route-aware correction, and the final velocity becomes closer to a controlled interpolation between the Driver direction and the route direction.
3. When $0 < \widetilde{s}_j < 1$, the model does not perform a discrete "on/off" selection, but instead carries out a continuous transition between the Driver base velocity and the Navigator route refinement.

Therefore, in the main text, the column-wise soft selector continuously models the correction suitability of different numerical columns, while the gating vector g dynamically adjusts the actual injection strength of local refinement according to the current sample state and time step. In other words, $\widetilde{s}$ models column-level adaptability, whereas g models local dynamic refinement strength. Second, the origin of Eq. (14) in the main text can be explained by a simple endpoint-consistency constraint. Suppose that after time t, the current state x_t continues to evolve with a constant velocity v_{route}, and is required to reach the endpoint $\widehat{x}_1$ predicted by the Navigator at $t = 1$. Then the following condition must hold:

$$x_t + (1 - t)v_{\mathrm{route}} = \widehat{x}_1 \tag{31}$$

Rearranging the above equation yields

$$v_{\mathrm{route}} = \frac{\widehat{x}_1 - x_t}{1 - t} \tag{32}$$

Therefore, the v_{route} in the main text is not an additional heuristic term, but rather the natural consequence of the constraint that the current state should reach the predicted endpoint with a constant velocity. In implementation, the denominator is written as $\max(1 - t, \varepsilon)$ solely to avoid numerical instability when t is close to 1.

B.2 Objective Functions

In this section, we provide additional clarification of the training objective presented in the main text. As described there, the overall loss is centered on the discrepancy between the final velocity field and the target velocity, while also including auxiliary constraints for the Driver, the Navigator, and the column-wise soft selector. Rather than restating the formal definitions, we focus here on the functional role and necessity of each component. First, the main velocity loss directly constrains the final velocity field $v_{\mathrm{final}}(x_t, t)$ in Eq. (16) of the main text to approximate the target velocity. As a result, its gradients are propagated jointly to the shared backbone, the Driver branch, the Navigator branch, the column-wise soft selector, and the gate module. In other words, this term is responsible for optimizing the structured velocity field as a whole, and serves as the core supervisory signal shared by all modules. Precisely for this reason, however, without additional constraints, the Driver branch may degenerate into merely an intermediate component that cooperates with the Navigator, while its independent ability to act as a global base velocity field could be weakened. To prevent such degeneration, the main text further introduces an auxiliary supervision term for the Driver, which directly constrains v_{drv} itself to approximate the target velocity. The purpose of this term is not to replace the main loss, but to explicitly ensure that the Driver alone possesses a usable global transport capability. In this way, even when the column-wise soft selector takes small values on some numerical dimensions, or when the gate output is weak at certain time steps, the model can still rely on v_{drv} to provide a stable base generation direction. For the Navigator, the endpoint reconstruction supervision and the correlation-preserving supervision are no longer applied only to a fixed set of activated dimensions, but are instead continuously weighted by the column-wise soft selector. That is, the model adaptively adjusts the relative importance of different numerical dimensions in endpoint reconstruction and local structure preservation according to the magnitude of s. This design is consistent with Eqs. (11) and (16) in the main text: if a numerical dimension has stronger column-wise adaptability, then it not only receives a larger weight in endpoint interpolation and final velocity fusion, but is also subject to stronger auxiliary supervision; conversely, its supervision strength is correspondingly reduced. Finally, the column-wise soft selector itself requires additional regularization. To this end, the main text further introduces a sparsity regularizer and a binarization-oriented regularizer. The former suppresses the trivial case in which all numerical columns receive excessively large refinement weights simultaneously, thereby encouraging the model to allocate stronger route-aware correction only to dimensions where it is truly needed. The latter prevents the selector from remaining in an uninformative intermediate regime for an extended period, and instead encourages the model to learn a clearer column-wise adaptability structure. Therefore, in the current formulation, the overall training objective of DN-Flow learns not only the structured velocity field itself, but also the column-wise adaptability representation together with the constraints needed to stabilize it.

B.3 Why Discrete Categorical Columns Do Not Participate in Navigator Refinement

Consider the one-hot encoding block corresponding to a categorical feature. Suppose this categorical feature has K possible values. Then its true endpoint satisfies

$$x_1^{(\mathrm{c})} \in \mathcal{V}_K := \{e_1, \ldots, e_K\} \subset \mathbb{R}^K, \quad \sum_{k=1}^{K} x_{1,k}^{(\mathrm{c})} = 1, \quad x_{1,k}^{(\mathrm{c})} \in \{0, 1\}. \tag{33}$$

If the Navigator mechanism were applied directly to this categorical block, let the corresponding endpoint prediction be denoted by $\widehat{x}_1^{(\mathrm{c})}$, while still using the same endpoint regression form as for continuous numerical dimensions. If the endpoint supervision is given by squared error, then the

corresponding conditional risk is

$$L_{\mathrm{rec}}^{(\mathrm{c})}\left(\widehat{x}_1^{(\mathrm{c})}\right) = \mathbb{E}\left[\left\|\widehat{x}_1^{(\mathrm{c})} - x_1^{(\mathrm{c})}\right\|_2^2 \mid h\right] \tag{34}$$

where h denotes the shared representation. Taking the derivative with respect to $\widehat{x}_1^{(\mathrm{c})}$ and setting it to zero gives

$$\frac{\partial L_{\mathrm{rec}}^{(\mathrm{c})}}{\partial \widehat{x}_1^{(\mathrm{c})}} = 2\left(\widehat{x}_1^{(\mathrm{c})} - \mathbb{E}[x_1^{(\mathrm{c})} \mid h]\right) = 0 \tag{35}$$

from which the optimal solution is

$$\widehat{x}_1^{(\mathrm{c})*} = \mathbb{E}[x_1^{(\mathrm{c})} \mid h] \tag{36}$$

Since $x_1^{(\mathrm{c})}$ is a one-hot variable, we have

$$\mathbb{E}[x_1^{(\mathrm{c})} \mid h] = (p_1(h), \ldots, p_K(h)), \quad p_k(h) \geq 0, \quad \sum_{k=1}^{K} p_k(h) = 1 \tag{37}$$

That is, under squared error, the optimal endpoint prediction does not lie on the vertex set $\mathcal{V}_K$, but rather in the interior of the probability simplex

$$\Delta^{K-1} := \left\{ z \in \mathbb{R}^K \mid z_k \geq 0,\ \sum_{k=1}^{K} z_k = 1 \right\} \tag{38}$$

Unless the conditional distribution fully collapses to a single category, that is, unless some $p_k(h) = 1$, we generally have

$$\widehat{x}_1^{(\mathrm{c})*} \in \Delta^{K-1} \setminus \mathcal{V}_K \tag{39}$$

Here, $\setminus$ denotes set difference. Therefore, if the same endpoint regression used for continuous numerical dimensions is directly applied to a categorical block, its optimal solution naturally corresponds to a probability vector rather than a discrete vertex. Furthermore, if we continue to use the route velocity definition in the main text, then the route velocity for the categorical block becomes

$$v_{\mathrm{route}}^{(\mathrm{c})} = \frac{\widehat{x}_1^{(\mathrm{c})} - x_t^{(\mathrm{c})}}{\max(1-t, \varepsilon)} \tag{40}$$

Since $x_t^{(\mathrm{c})}$ itself also lies in a continuous space, this route refinement drives a continuous vector toward another continuous vector, rather than representing a jump from one discrete vertex to another. After sampling is completed, the categorical block must still be projected back to a one-hot output through a discrete projection. Let this projection be denoted by

$$\Pi(z) = e_{\arg\max_k z_k} \tag{41}$$

Accordingly, the final categorical prediction actually used is $\Pi(\widehat{x}_1^{(\mathrm{c})})$, rather than $\widehat{x}_1^{(\mathrm{c})}$ itself. The problem is that $\Pi(\cdot)$ is a discontinuous mapping. Define

$$\mathcal{B} = \left\{ z \in \mathbb{R}^K \mid \exists\, i \neq j,\ z_i = z_j = \max_{\ell} z_\ell \right\} \tag{42}$$

as the projection boundary set. On $\mathcal{B}$, an arbitrarily small continuous perturbation may cause the discrete output to flip. More specifically, there exist two vectors $z, z' \in \mathbb{R}^K$ such that

$$\|z - z'\|_2 \ll 1, \quad \Pi(z) \neq \Pi(z') \tag{43}$$

For example, when $K = 2$, let $z = (0.51, 0.49)$ and $z' = (0.49, 0.51)$. Then $\|z - z'\|_2 = \sqrt{2} \times 0.02$, $\Pi(z) = e_1$, $\Pi(z') = e_2$. Therefore, a small value of $\left\|\widehat{x}_1^{(\mathrm{c})} - x_1^{(\mathrm{c})}\right\|_2$ does not imply that $\Pi(\widehat{x}_1^{(\mathrm{c})}) = x_1^{(\mathrm{c})}$. In other words, a small error in continuous space does not guarantee a correct discrete categorical prediction. This shows that the Navigator mechanism of "endpoint regression $\rightarrow$ route velocity $\rightarrow$ continuous refinement" is mathematically more suitable for numerical dimensions in continuous Euclidean space, but not for one-hot categorical blocks. Therefore, in Eq. (16) of the main text, we set all categorical dimensions to zero through the extended selection vector $\widetilde{s}$, i.e., $\widetilde{s}^{(\mathrm{c})} = 0$, so that categorical columns are always generated along the Driver path and do not directly receive endpoint-aware refinement from the Navigator. This theoretical mismatch is also supported by an empirical counterfactual study reported in Appendix E.3, where extending Navigator to categorical one-hot dimensions consistently degrades performance.

B.4 Loss Function Design

Our training objective consists of three main components: a final velocity field fitting loss, a supervision loss for the Driver's base velocity, and endpoint consistency and correlation-preserving losses for the Navigator.

1). Final Velocity Field Loss. The final velocity field is expected to approximate the target velocity $x_1 - x_0$. We therefore define:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{x_1,x_0,t}\left[\|v_{\text{final}}(x_t,t) - (x_1 - x_0)\|_2^2\right] \tag{44}$$

2). Driver Supervision Loss. To ensure that the Driver itself possesses sufficiently strong standalone modeling capability, we impose an additional supervision loss on the base velocity field:

$$\mathcal{L}_{\text{drv}} = \mathbb{E}_{x_1,x_0,t}\left[\|v_{\text{drv}} - (x_1 - x_0)\|_2^2\right] \tag{45}$$

This allows the model to retain stable global generation capability even when the Navigator refinement is weak or completely disabled.

3). Navigator Endpoint Reconstruction Loss. In DN-Flow, the column-wise selection mechanism imposes a soft-weighted supervision over all numerical dimensions. Let $\widehat{x}_{1,j}^{\text{num}}$ and $x_{1,j}^{\text{num}}$ denote the predicted endpoint and the ground-truth endpoint of the j-th numerical dimension, respectively. The endpoint reconstruction loss is defined as

$$\mathcal{L}_{x_1} = \frac{\sum_{j=1}^{d_{\text{num}}} s_j \, \mathbb{E}_{x_1}\left[\left(\widehat{x}_{1,j}^{\text{num}} - x_{1,j}^{\text{num}}\right)^2\right]}{\sum_{j=1}^{d_{\text{num}}} s_j + \varepsilon} \tag{46}$$

This definition is consistent with Eq. (11): the larger the column-wise soft selector s_j, the higher the weight assigned to the corresponding numerical dimension in the endpoint reconstruction supervision; conversely, when s_j is small, the endpoint supervision on that dimension is correspondingly weakened.

4). Correlation-Preserving Loss. To further preserve local correlation structure in the numerical part, we compare, at the batch level, the correlation matrices of the predicted endpoints and the ground-truth endpoints in the numerical space. Let $\widehat{X}_1^{\text{num}} \in \mathbb{R}^{B\times d_{\text{num}}}$ and $X_1^{\text{num}} \in \mathbb{R}^{B\times d_{\text{num}}}$ denote the numerical parts of the predicted endpoints and the real endpoints in a batch, respectively. We define the column-wise weight matrix as

$$W = ss^\top \tag{47}$$

where $W \in [0,1]^{d_{\text{num}}\times d_{\text{num}}}$ is formed by the outer product of the column-wise soft selector. The correlation-preserving loss is then written as

$$\mathcal{L}_{\text{corr}} = \frac{\left\| W \odot \left(\text{Corr}\left(\widehat{X}_1^{\text{num}}\right) - \text{Corr}(X_1^{\text{num}})\right)\right\|_F^2}{\|W\|_1 + \varepsilon} \tag{48}$$

The objective function continuously weights all pairs of numerical columns through W. If both s_i and s_j are large for two numerical columns, then the corresponding correlation term receives a larger weight in the loss. Conversely, if either column has low column-level adaptability, then the influence of that column pair in the correlation constraint is correspondingly reduced. This makes local correlation preservation consistent with the learning of column-wise adaptability.

5). Overall Objective. In addition to the main loss and auxiliary supervision terms above, we further introduce two lightweight regularizers to prevent the column-wise soft selector from degenerating into the trivial solution where "all numerical dimensions are fully activated" or from remaining in an uninformative intermediate regime for an extended period. First, we define a sparsity regularization term as

$$\mathcal{L}_{\text{sparse}} = \frac{1}{d_{\text{num}}} \sum_{j=1}^{d_{\text{num}}} s_j \tag{49}$$

Its role is to suppress overly large refinement weights being assigned to all numerical columns simultaneously, thereby encouraging the model to allocate stronger route-aware correction only to dimensions where it is truly needed. Second, we define a binarization-oriented regularization term as

$$\mathcal{L}_{\text{binary}} = \frac{1}{d_{\text{num}}} \sum_{j=1}^{d_{\text{num}}} s_j(1 - s_j) \tag{50}$$

This term is small when s_j is close to 0 or 1, and large when s_j is close to 0.5. It therefore encourages the column-wise soft selector to learn a clearer column-level adaptability structure, rather than remaining in an ambiguous intermediate state. Combining all terms above, the final optimization objective is written as

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_{\text{drv}}\mathcal{L}_{\text{drv}} + \lambda_{x_1}\mathcal{L}_{x_1} + \lambda_{\text{corr}}\mathcal{L}_{\text{corr}} + \lambda_{\text{sp}}\mathcal{L}_{\text{sparse}} + \lambda_{\text{bin}}\mathcal{L}_{\text{binary}} \tag{51}$$

This objective shows that the training of DN-Flow is no longer limited to learning a structured velocity field alone, but also jointly learns the column-wise adaptability representation s. Specifically, $\mathcal{L}_{\text{flow}}$ and $\mathcal{L}_{\text{drv}}$ supervise the final velocity field and the Driver's base velocity, respectively; $\mathcal{L}_{x_1}$ and $\mathcal{L}_{\text{corr}}$ constrain the Navigator in terms of endpoint reconstruction and local structure preservation in the numerical space; and $\mathcal{L}_{\text{sparse}}$ together with $\mathcal{L}_{\text{binary}}$ ensures that the column-wise soft selector learns a more stable and discriminative pattern of column-level adaptability.

B.5 Sampling Process

During sampling, we first initialize $x^{(0)} \sim \mathcal{N}(0, I)$. The time interval $[0, 1]$ is then uniformly divided into K steps, with step size $\Delta t = 1/K$. At the k-th step, we first compute the current velocity at $t_k = k/K$:

$$v_1 = v_{\text{final}}(x^{(k)}, t_k) \tag{52}$$

This yields the Euler prediction point

$$\widetilde{x} = x^{(k)} + \Delta t\, v_1 \tag{53}$$

If the process has not yet reached the final step, we compute the velocity once more at the next time point t_{k+1}:

$$v_2 = v_{\text{final}}(\widetilde{x}, t_{k+1}) \tag{54}$$

and apply the second-order Heun correction:

$$x^{(k+1)} = x^{(k)} + \frac{\Delta t}{2}(v_1 + v_2) \tag{55}$$

After all K steps are completed, the final synthetic sample $x^{(K)}$ is obtained. The current implementation follows exactly this second-order predictor–corrector sampling procedure, and thus each step typically involves two network evaluations. After continuous-space generation is completed, the model further performs numerical inverse transformation, hardening within categorical one-hot groups, column-order restoration, and data type recovery in sequence, ultimately producing synthetic tables with the same column structure and data types as the real data.

C Training and Sampling Details

In this section, we provide additional details of the training and sampling protocols that were not elaborated in the main text. To avoid repetition with Section 3, we do not reintroduce the continuous-time generative path or the structured velocity field itself, and instead focus only on the implementation details relevant to reproduction.

Algorithm 1 DN-Flow Training

Require: training distribution p_{data}, model parameters θ, selector parameter β
1: **repeat**
2: sample $x_1 \sim p_{\text{data}}$, $x_0 \sim \mathcal{N}(0, I)$, $t \sim \mathcal{U}(0, 1)$
3: construct $x_t = tx_1 + (1 - t)x_0$
4: compute h, v_{drv}, $\hat{x}_1$, v_{route}, g, and v_{final}
5: compute the six-term objective $\mathcal{L}$
6: update θ and β by gradient descent
7: **until** convergence

Algorithm 2 DN-Flow Sampling

Require: number of integration steps K
sample $x^{(0)} \sim \mathcal{N}(0, I)$; set $\Delta t = 1/K$
for $k = 0, \ldots, K-1$ **do**
  $t_k = k/K$, $v_1 = v_{\text{final}}(x^{(k)}, t_k)$
  $x_{\text{e}} = x^{(k)} + \Delta t\, v_1$
  if $k < K-1$ **then**
    $v_2 = v_{\text{final}}(x_{\text{e}}, t_{k+1})$
    $x^{(k+1)} = x^{(k)} + \frac{\Delta t}{2}(v_1 + v_2)$
  else
    $x^{(K)} = x_{\text{e}}$
  end if
end for
inverse-transform numerical columns; harden one-hot groups; recover data types
return synthetic table

C.1 Training Details

The training stage uses mini-batch stochastic optimization. For each batch, we first sample a batch of real training examples, and then independently draw, for each sample, a time variable and a Gaussian noise sample of the same dimensionality. The intermediate state is then constructed according to the continuous-time interpolation path defined in the main text, from which the corresponding target velocity is obtained. The purpose of this design is to randomly cover different intermediate states over the entire time interval, so that the model learns the full continuous transport field rather than fitting the mapping at only a single fixed time point. During the forward pass, the model takes the current intermediate state and time variable as input, and outputs the base velocity, endpoint estimate, route velocity, gated refinement, and final velocity field defined in the main text. The loss function adopts a joint form consisting of a main loss plus auxiliary losses. The main loss is always imposed on the final velocity field and constrains the overall continuous transport direction. The auxiliary Driver term is used to stabilize the learning of the base velocity, while the Navigator-related terms are continuously weighted by the column-wise soft selector, so as to impose adaptive supervision on endpoint reconstruction and local correlation preservation over the numerical part. Model parameters are updated using a gradient-based stochastic optimizer, and model selection is performed according to validation-set performance. During training, the validation set is used only for evaluation and checkpoint selection, and does not participate in parameter updates. The final experimental model is chosen as the one with the best validation loss. In other words, our model selection criterion is the optimization objective on the validation stage, rather than a post hoc choice based directly on Shape, Trend, or downstream-task metrics on the test set.

C.2 Sampling Details

The sampling stage starts from standard Gaussian noise initialization and progressively evolves from $t = 0$ to $t = 1$ along the velocity field defined in the main text. In implementation, the continuous time interval $[0, 1]$ is uniformly discretized into K time steps, and numerical integration is performed using a predictor–corrector scheme. Specifically, at the k-th time step t_k, we first compute the velocity field from the current state and perform an Euler prediction:

$$\widetilde{x}_{k+1} = x_k + \Delta t\, v_\theta(x_k, t_k) \tag{56}$$

where $\Delta t = 1/K$. Then, at the predicted state $\widetilde{x}_{k+1}$, we compute the velocity at the next time point again and apply the Heun correction:

$$x_{k+1} = x_k + \frac{\Delta t}{2}\left[v_\theta(x_k, t_k) + v_\theta(\widetilde{x}_{k+1}, t_{k+1})\right] \tag{57}$$

Therefore, our sampling stage does not use the simplest first-order Euler integration, but instead adopts a more stable predictor–corrector numerical integration scheme to reduce the error introduced by discrete-time approximation. For the final time step, since there is no subsequent time point for correction, the final predicted state can be directly taken as the output.

Table 8: Hyperparameter settings used to train DN-Flow on each dataset.

Datasets	λ_{x_1}	λ_{corr}	λ_{drv}	λ_{sp}	λ_{bin}	α
Adult	0.10	0.05	0.10	1×10^{-3}	1×10^{-3}	0.05
Default	0.15	0.05	0.10	1×10^{-3}	1×10^{-5}	0.10
Shoppers	0.20	0.05	0.05	1×10^{-3}	1×10^{-3}	0.05
Magic	0.20	0.15	0.20	1×10^{-3}	1×10^{-5}	0.10
Beijing	0.10	0.05	0.10	1×10^{-3}	1×10^{-3}	0.05
News	0.20	0.05	0.10	1×10^{-5}	1×10^{-3}	0.10
BM	0.20	0.10	0.10	1×10^{-3}	1×10^{-3}	0.10
BS	0.20	0.05	0.05	1×10^{-3}	1×10^{-3}	0.05

C.3 Output Recovery and Post-processing

After sampling, the model output is still a continuous representation in the preprocessed space, rather than a directly usable sample in the original tabular domain. Therefore, after obtaining the final continuous output, an inverse preprocessing step is required to map it back to the original table space. For numerical features, we first apply the inverse transformation of the numerical transformer used during training, so as to recover the original numerical scale. For categorical features, since they are expanded into one-hot representations before training, the recovery stage first requires a discretization step within each categorical group, that is, keeping only the position of the maximum component and setting all other components to zero, after which the one-hot representation is mapped back to the original categorical value. Only after this procedure can the generated sample be regarded as a synthetic instance in the original tabular space. Although this recovery process is not part of the velocity field definition itself, it is crucial for experimental evaluation. This is because all of our Shape, Trend, MLE, and visualization analyses are ultimately conducted on samples recovered in the original semantic space, rather than directly in the model's internal continuous representation space.

C.4 Additional Notes for Reproducibility

Finally, we provide several implementation conventions relevant to reproducibility that are not suitable for elaboration in the main text. First, the training and validation stages are kept strictly separate, and the validation set is used only for model selection. Second, the weights of the auxiliary loss terms all affect the final sample quality, and the corresponding hyperparameters are therefore reported together in Appendix D. Third, the inverse transformation of categorical features depends on the one-hot group definitions established during preprocessing. Hence, the preprocessing and inverse-preprocessing mappings must remain strictly consistent across different datasets; otherwise, valid discrete outputs cannot be recovered.

D Implementation Details

D.1 Data Preprocessing and Inverse Transform

For all datasets, we first determine the feature order, target column, and numerical/categorical partition according to the dataset metadata, and then apply a unified preprocessing pipeline following (Shi et al., 2025). Numerical columns are transformed using `QuantileTransformer` to obtain approximately Gaussian marginals, while categorical columns are represented by one-hot encoding. All preprocessors are fitted only on the training split and are reused during generation for inverse transformation back to the original tabular space.

D.2 Architecture and Hyperparameters

Except for dataset-specific loss weights and the global correction coefficient, all eight datasets share the same model architecture and training configuration. DN-Flow uses a hidden dimension of 512, six TransformerEncoder blocks, eight attention heads, a feed-forward hidden dimension of 1024, $K = 100$ sampling steps, a batch size of 512, and 1000 training epochs. Table 8 reports the dataset-specific values of λ_{x_1}, λ_{corr}, λ_{drv}, λ_{sp}, λ_{bin}, and α.

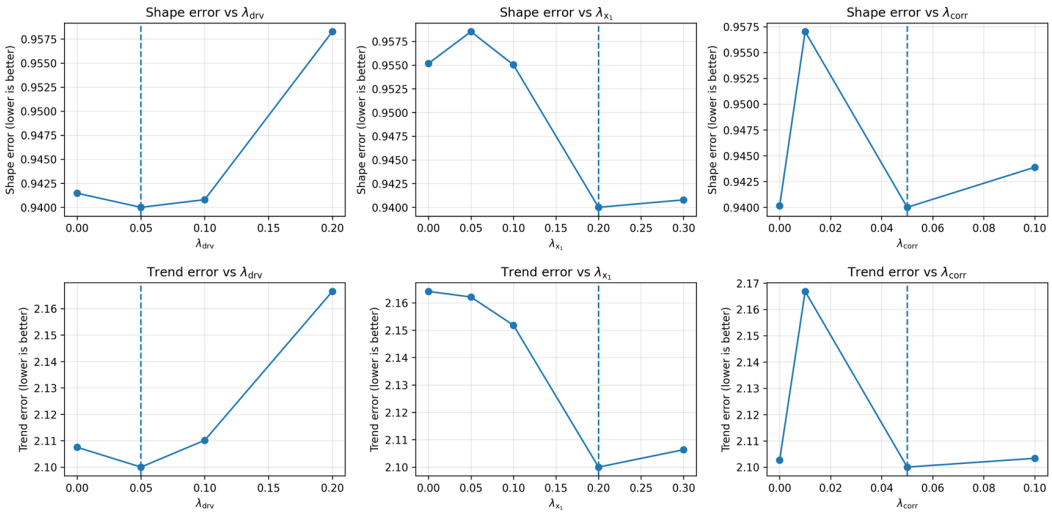


Figure 5: Sensitivity analysis of λ_{drv}, λ_{x_1}, and λ_{corr}.

D.3 Hyperparameter Selection and Sensitivity

The same candidate grids and fixed-order coordinate-wise search procedure were used across all eight datasets. Hyperparameter selection used only the training and validation splits. Each candidate was first evaluated with one training seed and three sampling seeds, after which the strongest candidates were re-evaluated with three training seeds and five sampling seeds per checkpoint. Sampling results were averaged within each training seed, and selection was based on the average normalized validation Shape and Trend errors. The test split was not used during hyperparameter selection. Baselines used their official or recommended per-dataset configurations.

Figures 5–7 illustrate representative sensitivity analyses on the Shoppers dataset. Figure 5 shows the effects of λ_{drv}, λ_{x_1}, and λ_{corr}. Moderate auxiliary supervision generally performs best: $\lambda_{\text{drv}} = 0.05$, $\lambda_{x_1} = 0.20$, and $\lambda_{\text{corr}} = 0.05$ jointly provide the lowest Shape and Trend errors on Shoppers, whereas overly weak or strong weights degrade performance.

Figure 6 examines the selector regularization weights. Both λ_{sp} and λ_{bin} achieve their best performance around 10^{-3} on Shoppers. Deviating substantially from this scale increases both Shape and Trend errors, indicating that the selector benefits from a moderate balance between sparsity and decisiveness.

Figure 7 evaluates the global correction coefficient α. The best Shoppers result is obtained at $\alpha = 0.05$. Setting $\alpha = 0$ removes the explicit Navigator route correction and slightly degrades both metrics, while larger correction strengths provide no systematic benefit. This supports using the Navigator as a controlled local refinement mechanism rather than allowing it to dominate the Driver transport.

D.4 Training Configuration, Checkpoint Selection, and Reproducibility

We use `AdamW` for optimization with an initial learning rate of 2×10^{-4} and a `ReduceLROnPlateau` scheduler. If the validation loss does not improve for 20 epochs, the learning rate is reduced by 50%, down to a minimum of 1×10^{-6}. Gradient clipping is applied for additional training stability.

All experiments were conducted on an NVIDIA GeForce RTX 4070 GPU. Training and validation splits follow (Shi et al., 2025). Checkpoint selection is based exclusively on validation loss: whenever the validation loss improves, the corresponding model is saved as the current best checkpoint. Test-set metrics are not used for model selection. For final evaluation, each model is trained with five training seeds, and 20 sampling seeds are evaluated for the best checkpoint of each run. The generated sample size is always matched to the size of the corresponding real training set.

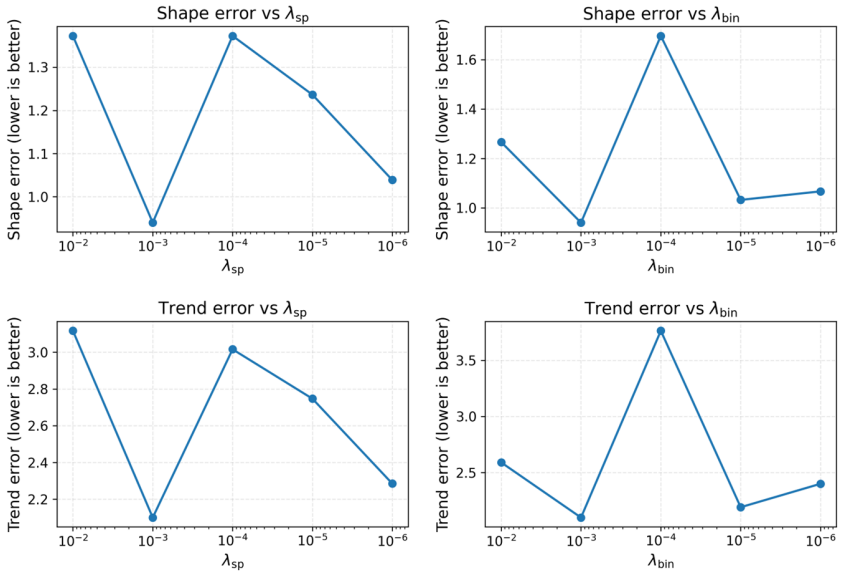


Figure 6: Sensitivity analysis of λ_{sp} and λ_{bin}.

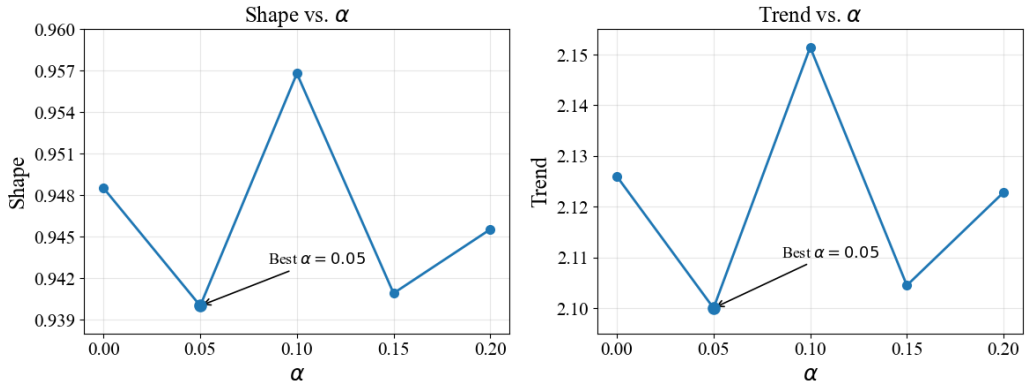


Figure 7: Sensitivity analysis of α.

D.5 Visualization of Column-wise Adaptability

To illustrate the behavior of the column-wise selector, Figure 8 visualizes the learned selector scores on Adult. Numerical columns such as "hours.per.week", "fnlwgt", and "education.num" receive relatively larger selector values, whereas "capital.gain" and "capital.loss" receive smaller values. This shows that DN-Flow does not apply the same refinement strength uniformly across numerical columns.

The selector should not, however, be interpreted as a conventional feature-importance score. Its effect on the final correction also depends on the predicted endpoint, the dynamic gate, and the discrepancy between Driver and route velocities. Cross-seed selector stability and its relationship to column-wise fidelity gains are analyzed separately in Appendix E.

E Additional Experimental Results

This appendix provides supplementary analyses beyond the main Shape, Trend, and MLE results, including additional fidelity metrics, type-wise fidelity, privacy diagnostics, categorical-Navigator ablations, selector stability, statistical reliability, and timestep-dependent Gate behavior.

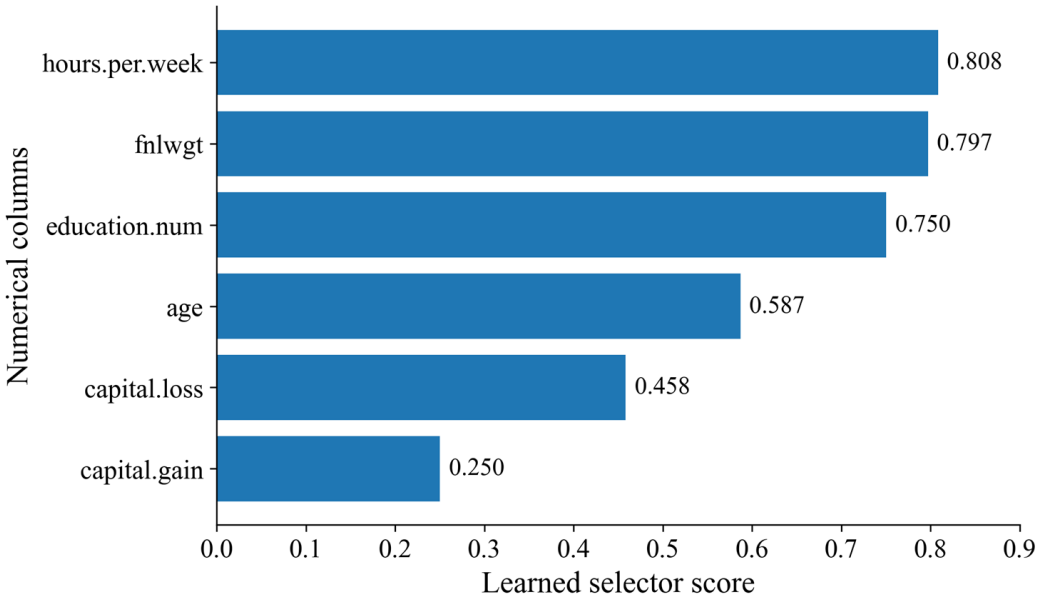


Figure 8: Visualization of learned selector scores on the Adult dataset.

Table 9: Comparison of α-Precision scores.

Methods	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
CTGAN	77.74±0.15	62.08±0.08	76.97±0.39	86.90±0.22	96.27±0.14	96.96±0.17	88.45±0.21	90.12±0.35	84.44
TVAE	98.17±0.17	85.57±0.34	58.19±0.26	86.19±0.48	97.20±0.10	86.41±0.17	87.34±0.25	84.15±0.41	85.40
GOGGLE	50.68	68.89	86.95	90.88	88.81	86.41	82.46	84.77	79.98
GReaT	55.79±0.03	85.90±0.17	78.88±0.13	85.46±0.54	98.32±0.22	OOM	87.45±0.32	88.12±0.45	82.85
STaSy	82.87±0.26	90.48±0.11	89.65±0.25	86.56±0.19	89.16±0.12	94.76±0.33	91.25±0.18	92.68±0.27	89.68
CoDi	77.58±0.45	82.38±0.15	94.95±0.35	85.01±0.36	98.13±0.38	87.15±0.12	89.14±0.31	90.55±0.22	88.11
TabDDPM	96.36±0.20	97.59±0.36	88.55±0.68	98.59±0.17	97.93±0.30	0.00±0.00	25.94±0.21	22.61±0.26	65.95
TABSYN	**99.39±0.18**	98.65±0.23	98.36±0.52	99.42±0.28	97.51±0.24	95.05±0.30	98.50±0.18	96.59±0.12	97.93
TABDIFF	99.02±0.20	98.49±0.28	**99.11±0.34**	99.47±0.21	98.06±0.24	97.36±0.17	98.88±0.17	**99.41±0.20**	98.73
DN-Flow	98.88±0.16	**98.80±0.21**	98.82±0.31	**99.59±0.18**	**99.14±0.20**	**97.85±0.20**	**99.44±0.15**	99.05±0.22	**98.95**

E.1 Additional Fidelity Metrics

In addition to Shape and Trend, we evaluate synthetic-data fidelity using α-Precision, β-Recall, and the Detection score (C2ST), which respectively characterize sample realism, distributional coverage, and real–synthetic distinguishability.

α-Precision. Table 9 reports the α-Precision results. DN-Flow achieves the highest average score of 98.95, compared with 98.73 for TABDIFF and 97.93 for TABSYN, and obtains the best result on five of the eight datasets.

β-Recall. Table 10 reports distributional coverage. DN-Flow obtains the highest average β-Recall of 50.87, compared with 50.47 for TABDIFF and 48.17 for TABSYN, while achieving the best result on six datasets. Taken together, Tables 9 and 10 indicate that DN-Flow maintains strong sample realism while retaining competitive distributional coverage.

Detection Score (C2ST). Table 11 reports C2ST-based Detection scores, where higher values indicate lower real–synthetic distinguishability under this evaluation. DN-Flow obtains the highest average score of 0.9821.

E.2 Type-wise Fidelity Decomposition

Because the Navigator operates directly only on numerical dimensions, we further decompose the existing Shape and Trend evaluations by feature type. Table 12 reports numerical and categorical Shape separately, together with numerical–numerical (NN), categorical–categorical (CC), and numerical–categorical (NC) pairwise dependency errors. Each entry compares Base with DN-Flow using the same generated samples and evaluation protocol as in the main experiments.

Table 10: Comparison of β-Recall scores.

Methods	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
CTGAN	30.80±0.20	18.22±0.17	31.80±0.35	11.75±0.20	34.80±0.10	24.97±0.29	22.45±0.25	32.14±0.30	25.87
TVAE	38.87±0.31	23.13±0.11	19.78±0.10	32.44±0.35	28.45±0.08	29.66±0.21	26.50±0.22	33.75±0.28	29.07
GOGGLE	8.80	14.38	9.79	9.88	19.87	2.03	8.95	13.42	10.89
GReaT	49.12±0.18	42.04±0.19	44.90±0.17	34.91±0.28	43.34±0.31	OOM	40.15±0.24	47.85±0.32	43.19
STaSy	29.21±0.34	39.31±0.39	37.24±0.45	**53.97±0.57**	54.79±0.18	39.42±0.32	41.25±0.35	52.10±0.40	43.41
CoDi	9.20±0.15	19.94±0.22	20.82±0.23	50.56±0.31	52.19±0.12	34.40±0.31	31.50±0.28	46.25±0.35	33.11
TabDDPM	47.05±0.25	47.83±0.35	47.79±0.25	48.46±0.42	56.92±0.13	0.00±0.00	0.00±0.00	0.46±0.01	31.06
TABSYN	47.92±0.23	46.45±0.35	49.10±0.60	48.03±0.50	59.15±0.22	43.01±0.28	41.96±0.24	49.72±0.20	48.17
TABDIFF	51.64±0.20	51.09±0.25	49.75±0.64	48.01±0.31	**59.63±0.23**	42.10±0.32	**46.28±0.22**	55.22±0.17	50.47
DN-Flow	**52.79±0.20**	**52.81±0.20**	**49.81±0.45**	49.62±0.27	58.93±0.16	**43.18±0.22**	43.44±0.18	**56.37±0.15**	**50.87**

Table 11: Detection score (C2ST) using logistic regression classifier.

Method	Adult	Default	Shoppers	Magic	Beijing	News	BM	BS	Average
CTGAN	0.5949	0.4875	0.7488	0.6728	0.7531	0.6947	0.6850	0.7120	0.6686
TVAE	0.6315	0.6547	0.2962	0.7706	0.8659	0.4076	0.6650	0.7010	0.6241
GOGGLE	0.1114	0.5163	0.1418	0.9526	0.4779	0.0745	0.2510	0.3150	0.3551
GReaT	0.5376	0.4710	0.4285	0.4326	0.6893	OOM	0.5850	0.6120	0.5366
STaSy	0.4054	0.6814	0.5482	0.6939	0.7922	0.5287	0.6620	0.5890	0.6126
CoDi	0.2077	0.4595	0.2784	0.7206	0.7177	0.0201	0.4550	0.3850	0.4055
TabDDPM	0.9755	0.9712	0.8349	**0.9998**	0.9513	0.0002	0.0003	0.1029	0.6045
TABSYN	0.9910	0.9826	0.9662	0.9960	0.9528	0.9255	0.9534	**0.9680**	0.9669
TABDIFF	0.9950	0.9774	**0.9843**	0.9989	0.9781	0.9308	0.9531	0.9010	0.9648
DN-Flow	**0.9963**	**0.9857**	0.9768	0.9931	**0.9951**	**0.9852**	**0.9801**	0.9443	**0.9821**

As shown in Table 12, numerical Shape, categorical Shape, and NN dependency errors improve on all eight datasets. CC improves on six of the seven applicable datasets, while NC improves on six of eight. Magic contains only one categorical column and therefore has no valid categorical–categorical pair; its CC entry is marked "—", and the CC average is computed over the remaining seven datasets. These results show that the fidelity gains are not confined to numerical marginals, although the Navigator's direct refinement remains specific to numerical dimensions.

E.3 Privacy Results

We use Distance to Closest Records (DCR) as an empirical diagnostic of whether synthetic samples are disproportionately close to the training set. A value near 50% indicates approximately balanced nearest-neighbor proximity to training and held-out data. DCR should be interpreted as a diagnostic rather than a formal privacy guarantee. Table 13 reports the DCR results. As shown in Table 13, DN-Flow remains relatively close to 50% across the evaluated datasets and does not exhibit the large deviations observed for some baselines. These results provide no indication of increased training-set memorization under this diagnostic.

E.4 Ablation on Extending Navigator to Categorical Dimensions

To test whether the same Navigator mechanism should be applied to categorical dimensions, we construct DN-Flow-AllNav, which extends continuous endpoint prediction and route correction to the one-hot categorical dimensions. Table 14 compares this variant with the original DN-Flow.

As shown in Table 14, extending the Navigator to categorical dimensions degrades both Shape and Trend on all eight datasets, increasing the averages from 0.849 to 0.980 and from 1.551 to 1.703, respectively. Numerical endpoints lie in a continuous Euclidean space, whereas one-hot categorical blocks correspond to discrete simplex vertices. Applying the same unconstrained continuous endpoint regression to the latter can therefore produce intermediate states poorly aligned with categorical geometry. These results support using the Driver for full mixed-type transport while restricting endpoint-guided Navigator refinement to numerical dimensions.

E.5 Selector Stability and Column-wise Fidelity Gains

We next examine whether the learned selector is stable across training seeds and whether larger selector values are associated with larger column-level fidelity gains. For each dataset, selector stability is measured using the ten pairwise Spearman rank correlations among the selector vectors obtained from the five training seeds. We also compute the Spearman correlation between selector

Table 12: Type-wise fidelity decomposition. Each cell reports Base → DN-Flow (lower is better).

Dataset	Numerical Shape	Categorical Shape	NN	CC	NC
Adult	0.440±0.036 → 0.433±0.034	0.350±0.028 → 0.345±0.027	0.641±0.121 → 0.609±0.118	1.049±0.123 → 1.039±0.120	1.423±0.124 → 1.374±0.121
Default	0.261±0.025 → 0.208±0.022	0.200±0.021 → 0.190±0.018	0.578±0.168 → 0.567±0.167	0.946±0.171 → 0.941±0.170	1.282±0.173 → 1.277±0.172
Shoppers	1.097±0.080 → 1.028±0.076	0.850±0.067 → 0.830±0.063	1.167±0.095 → 1.120±0.098	1.909±0.097 → 1.948±0.100	2.588±0.098 → 2.704±0.101
Magic	0.630±0.064 → 0.595±0.061	0.553±0.057 → 0.540±0.054	0.689±0.128 → 0.686±0.127	—	1.527±0.143 → 1.531±0.142
Beijing	0.627±0.033 → 0.593±0.032	0.500±0.028 → 0.490±0.027	1.010±0.096 → 0.888±0.088	1.652±0.098 → 1.586±0.090	2.239±0.099 → 2.050±0.091
News	2.148±0.021 → 2.003±0.020	1.500±0.019 → 1.450±0.018	1.113±0.041 → 1.052±0.039	1.821±0.042 → 1.766±0.040	2.468±0.051 → 2.373±0.049
Bank Marketing	0.906±0.068 → 0.807±0.056	0.700±0.054 → 0.670±0.045	0.901±0.078 → 0.847±0.069	1.475±0.080 → 1.453±0.070	2.000±0.081 → 1.935±0.070
Bike Sharing	1.770±0.096 → 1.640±0.084	1.350±0.072 → 1.310±0.063	1.650±0.069 → 1.518±0.059	2.700±0.070 → 2.646±0.060	3.660±0.070 → 3.485±0.060
Average	0.985±0.053 → 0.913±0.048	0.750±0.043 → 0.728±0.039	0.969±0.100 → 0.911±0.096	1.650±0.097 → 1.626±0.093	2.148±0.105 → 2.091±0.101

Table 13: DCR score, which quantifies how often a generated sample is closer to the training set than to the held-out test set. A value closer to 50% is preferable.

Method	Adult	Default	Shoppers	Beijing	News	BM	BS
STaSy	50.33%±0.19	50.23%±0.09	51.53%±0.16	50.59%±0.29	**50.59%±0.14**	50.85%±0.20	51.22%±0.25
CoDi	**49.92%±0.18**	51.82%±0.26	51.06%±0.18	50.87%±0.11	50.79%±0.23	51.68%±0.18	51.45%±0.22
TabDDPM	51.14%±0.18	52.15%±0.20	63.23%±0.25	80.11%±2.68	79.31%±0.29	46.97%±0.12	28.83%±0.25
TABSYN	50.94%±0.17	51.20%±0.18	52.90%±0.22	50.37%±0.13	50.85%±0.33	49.00%±0.20	52.99%±0.17
TABDIFF	50.10%±0.32	51.11%±0.36	50.24%±0.62	50.50%±0.36	51.04%±0.32	49.18%±0.23	49.14%±0.20
DN-Flow	49.40%±0.10	**50.21%±0.08**	**49.78%±0.16**	**50.09%±0.11**	48.25%±0.21	**49.25%±0.16**	**49.42%±0.12**

values and matched column-wise Shape improvements relative to Base. Table 15 summarizes the results.

As shown in Table 15, selector rankings are positively correlated across seeds for all datasets, with a macro average of 0.79. The selector–gain correlation is also positive on all eight datasets, averaging 0.60, and DN-Flow improves 91 of 104 numerical columns relative to Base. The relationship is not strictly monotonic, however, and the selector should not be interpreted as a conventional feature-importance score: the final correction also depends on the predicted endpoint, the sample- and time-dependent Gate, and the discrepancy between Driver and route velocities.

E.6 Statistical Reliability of the Backbone Comparison

Sampling variation is nested within each training seed; the 20 synthetic datasets generated from one checkpoint are not treated as independent training outcomes. Reported means and standard deviations are therefore computed after first averaging sampling-seed results within each of the five training seeds.

For the parameter-matched MLP comparison, we additionally use paired training seeds and report the mean difference, a 95% t-confidence interval over the five paired differences, and the number of matched-seed wins. Table 16 summarizes the results.

Table 16 shows that the paired mean favors DN-Flow in all 16 dataset–metric comparisons, while the 95% confidence interval excludes zero in 4 of the 16 cases. We therefore interpret the Transformer-block backbone as providing a modest and numerically consistent advantage rather than statistically reliable superiority on every dataset. More importantly, the parameter-matched MLP remains competitive, supporting the backbone-agnostic nature of the Driver–Navigator formulation.

E.7 Timestep-wise Analysis of the Gate

To examine how the Gate behaves along the flow trajectory, we performed a post-convergence analysis on all eight datasets. We divided $t \in [0, 1]$ into ten equal-width bins and evaluated 5,000 training samples per bin without updating model parameters. Because Driver and Navigator use different prediction targets, their original training losses are not directly comparable. We instead evaluate the velocity error of the Driver, the endpoint-induced route, and the final gated velocity on the numerical dimensions where Navigator operates. Selected timestep bins are reported in Table 17.

As shown in Table 17, the clearest behavior occurs near the endpoint. As $t \to 1$, errors in the predicted endpoint are amplified when converted into route velocity through the shrinking remaining-time denominator. In the final bin, Route MSE rises to 4.021, compared with 0.805 for Driver, while the raw Gate decreases to 0.228 and the effective correction weight to 0.099. The resulting Final MSE remains at 0.800. This suggests that the Gate primarily regulates correction magnitude rather than making a hard choice between the two branches, suppressing an increasingly sensitive endpoint-induced route near the end of the trajectory.

Table 14: Ablation results for extending the Navigator to categorical dimensions.

Datasets	Shape		Trend	
	DN-Flow	DN-Flow-AllNav	DN-Flow	DN-Flow-AllNav
Adult	**0.38±0.03**	0.41±0.03	**1.15±0.12**	1.29±0.15
Default	**0.20±0.02**	0.36±0.03	**1.00±0.17**	1.28±0.20
Shoppers	**0.94±0.07**	1.21±0.06	**2.10±0.10**	2.23±0.12
Magic	**0.59±0.06**	0.66±0.08	**0.84±0.13**	0.91±0.13
Beijing	**0.55±0.03**	0.60±0.02	**1.61±0.09**	1.68±0.10
News	**1.98±0.02**	2.08±0.02	**1.16±0.04**	1.22±0.06
BM	**0.73±0.05**	0.88±0.05	**1.60±0.07**	1.89±0.12
BS	**1.42±0.07**	1.64±0.07	**2.95±0.06**	3.12±0.08
Average	**0.849**	0.980	**1.551**	1.703

Table 15: Cross-seed selector stability and its relationship with numerical-column Shape improvements.

Dataset	Cross-seed selector rank corr. ↑	Selector–Shape gain corr. ↑	Improved numerical columns
Adult	0.86±0.07	0.83±0.11	6/6
Default	0.79±0.10	0.62±0.13	12/14
Shoppers	0.72±0.13	0.44±0.18	8/10
Magic	0.81±0.09	0.61±0.14	9/10
Beijing	0.85±0.07	0.71±0.11	7/7
News	0.75±0.11	0.53±0.09	39/46
Bank Marketing	0.82±0.08	0.64±0.12	6/7
Bike Sharing	0.68±0.16	0.40±0.20	4/4
Macro avg./total	0.79	0.60	91/104

F Efficiency and Convergence Analysis

This appendix analyzes the computational efficiency, sampling trade-off, and convergence behavior of DN-Flow. We first compare training and sampling time with representative diffusion-based baselines on the Adult dataset, and then examine the effect of the ODE solver and number of sampling steps on runtime and generation quality.

F.1 Training and Sampling Efficiency

Table 18 reports the training and sampling efficiency on Adult. Under the current implementation, a complete DN-Flow training run takes 22 minutes, compared with 112 minutes for TabDDPM, 85 minutes for TABSYN, and 94 minutes for TABDIFF. Thus, the additional Driver–Navigator heads and control mechanisms do not introduce a large training-time overhead. For sampling, the main experiments use a second-order Heun predictor–corrector solver with $K = 100$ integration steps. Each non-final Heun step evaluates the velocity field twice, giving a total of $2K - 1 = 199$ network function evaluations (NFEs). Under this configuration, DN-Flow requires 53.31 seconds, which is faster than TabDDPM but slower than TABSYN and TABDIFF. The higher sampling cost mainly results from the number of network evaluations required by the Heun trajectory.

F.2 Solver and NFE Trade-off

To examine the runtime–quality trade-off more directly, we vary only the ODE solver and number of integration steps while keeping the trained checkpoint, preprocessing, and sample count fixed. For each configuration, results are averaged over 20 sampling seeds. Table 19 reports the total trajectory NFE, sampling time, Shape, and Trend on Adult. As shown in Table 19, Heun consistently achieves lower Shape and Trend errors than Euler under nearly matched NFE budgets. For example, Heun-25 and Euler-50 use 49 and 50 NFEs, respectively, but obtain Shape/Trend errors of 0.39/1.16 and 0.50/1.30. Similarly, Heun-50 and Euler-100 require 99 and 100 NFEs, while achieving 0.38/1.15 and 0.44/1.22, respectively. Their runtimes are also nearly identical at matched NFE budgets, suggesting that network evaluations dominate the sampling cost rather than the numerical solver

Table 16: Paired statistical comparison between DN-Flow and the parameter-matched residual MLP backbone. Positive gains favor DN-Flow.

Dataset	Shape gain [95% CI]	Wins	Trend gain [95% CI]	Wins
Adult	+0.033 [+0.015,+0.051]	5/5	+0.069 [−0.001,+0.139]	4/5
Default	+0.008 [−0.003,+0.019]	4/5	+0.081 [−0.012,+0.174]	4/5
Shoppers	+0.022 [−0.018,+0.062]	4/5	+0.054 [−0.003,+0.111]	4/5
Magic	+0.006 [−0.027,+0.039]	3/5	+0.013 [−0.061,+0.087]	3/5
Beijing	+0.013 [−0.004,+0.030]	4/5	+0.114 [+0.063,+0.165]	5/5
News	+0.032 [+0.020,+0.044]	5/5	+0.023 [−0.001,+0.047]	4/5
Bank Marketing	+0.012 [−0.015,+0.039]	4/5	+0.028 [−0.011,+0.067]	4/5
Bike Sharing	+0.027 [−0.013,+0.067]	4/5	+0.062 [+0.027,+0.097]	5/5

Table 17: Selected timestep bins averaged over all eight datasets. Errors are computed on the numerical dimensions where Navigator operates.

t	Raw Gate	Effective Gate	Driver MSE	Route MSE	Final MSE
0.05	0.999	0.522	3.020	3.028	3.020
0.55	0.974	0.509	0.985	1.001	0.985
0.75	0.794	0.391	0.796	0.879	0.795
0.85	0.589	0.277	0.754	1.162	0.752
0.95	0.228	0.099	0.805	4.021	0.800

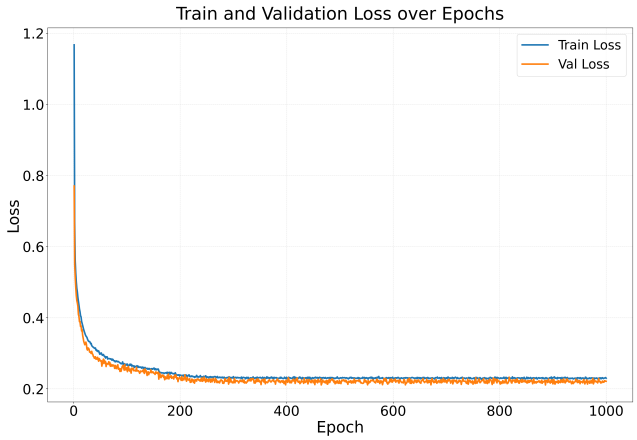


Figure 9: Training and validation losses of DN-Flow on the Adult dataset. Both losses decrease rapidly during early training and enter a relatively stable regime after approximately 200 epochs.

itself. Generation quality improves rapidly at low sampling budgets and then saturates. Increasing Heun from 5 to 10 steps reduces Shape from 0.69 to 0.43 and Trend from 1.58 to 1.21, whereas improvements become small after approximately 25–50 steps. In particular, Heun-50 reduces the total NFE from 199 to 99 and the runtime from 53.31 to 26.93 seconds while retaining the same Shape/Trend values as Heun-100. Heun-25 further reduces the total NFE to 49 and runtime to 13.35 seconds, with only a small increase from 0.38/1.15 to 0.39/1.16.

The main results in the paper retain the original Heun-100 configuration to preserve consistency with the submitted evaluation protocol. The lower-budget Heun-50 and Heun-25 configurations are therefore reported as alternative operating points rather than being used to retrospectively replace the main experimental results.

F.3 Training Convergence

Figure 9 shows the training and validation losses of DN-Flow on Adult over the training process. Both losses decrease rapidly during the early epochs and then decline more gradually, entering a relatively stable regime after approximately 200 epochs. The training and validation curves remain close throughout optimization, without a pronounced divergence. The convergence behavior indicates that

Table 18: Efficiency comparison on the Adult dataset. Train.T denotes the time for one complete training run, Sample.T denotes the sampling time, and NFE/step denotes the approximate number of network evaluations per integration step.

Method	Train.T (min)	Sample.T (sec)	NFE/step
TabDDPM	112	125.1±0.01	1
TABSYN	45+40=85	8.8±0.005	2
TABDIFF	94	15.2±0.007	2
DN-Flow	22	53.31±0.01	≈2

Table 19: Runtime–quality trade-off across ODE solvers and sampling budgets on Adult. Lower is better for Time, Shape, and Trend.

Solver	Steps	Total NFE	Time (s) ↓	Shape ↓	Trend ↓
Euler	5	5	1.43±0.21	1.78±0.15	3.58±0.39
Heun	5	9	2.48±0.13	0.69±0.05	1.58±0.17
Euler	10	10	2.75±0.08	1.02±0.08	2.19±0.23
Euler	15	15	4.12±0.04	0.81±0.06	1.79±0.19
Heun	10	19	5.21±0.03	0.43±0.03	1.21±0.12
Euler	25	25	6.83±0.03	0.63±0.05	1.50±0.16
Heun	15	29	7.92±0.01	0.41±0.03	1.18±0.12
Heun	25	49	13.35±0.01	0.39±0.03	1.16±0.12
Euler	50	50	13.60±0.00	0.50±0.04	1.30±0.14
Heun	50	99	26.93±0.02	0.38±0.03	1.15±0.12
Euler	100	100	26.95±0.01	0.44±0.04	1.22±0.12
Euler	125	125	34.05±0.01	0.43±0.04	1.21±0.12
Euler	150	150	40.92±0.01	0.42±0.03	1.20±0.12
Heun	100	199	53.31±0.01	0.38±0.03	1.15±0.12
Heun	125	249	67.88±0.00	0.38±0.03	1.15±0.11
Heun	150	299	81.60±0.01	0.38±0.03	1.15±0.11

most of the reduction in the optimization objective occurs during the earlier part of the 1000-epoch schedule, while later epochs mainly refine the solution around a relatively stable region. This suggests that the current training procedure is stable, while earlier stopping or more aggressive learning-rate scheduling may provide additional opportunities to reduce training cost.

G Limitations and Discussion

The current Navigator is specialized for numerical dimensions. Categorical variables are still modeled by the full-dimensional Driver and shared representation, but they do not receive the same endpoint-guided local refinement. This design is motivated by the mismatch between continuous endpoint regression and the discrete simplex geometry of one-hot categorical variables. The DN-Flow-AllNav experiment in Appendix E confirms that directly extending the same continuous refinement mechanism to categorical dimensions degrades average Shape from 0.849 to 0.980 and Trend from 1.551 to 1.703. Nevertheless, this does not imply that categorical variables cannot benefit from specialized refinement. A discrete-aware Navigator operating on category-group logits or simplex-compatible representations is a natural direction for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper clearly states its main claims in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper provides the mathematical formulation of the method and includes a theoretical/structural analysis. Please refer to appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper reports datasets, baselines, metrics, repeated-run protocol, main results, and ablation studies in the main paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The implementation of DN-Flow, together with instructions for data preparation, training, and synthetic data generation, is publicly available at `https://github.com/xmdz521-sudo/dn-flow`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Results are reported as mean ± standard deviation over repeated experiments, based on five training seeds and twenty sampling seeds for each best checkpoint unless otherwise specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on the methodological design and empirical evaluation of mixed-type tabular data generation rather than on a separate analysis of societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not involve a high-risk released system or dataset requiring special safeguards under this checklist item.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets such as datasets and code libraries used in the research are properly credited, and their licenses are respected and clearly mentioned.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The submission does not introduce a new dataset or other newly released asset requiring separate documentation under this checklist item.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work did not involve crowdsourcing or data collection from human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work did not involve human-subject studies requiring IRB approval or equivalent oversight.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology, experiments, and assets in this paper do not rely on LLM-generated content that requires declaration under this checklist item.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.