FLOWCF: SPARSE COUNTERFACTUAL EXPLANATIONS FOR MIXED-TYPE TABULAR DATA USING FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

In the field of Explainable AI (XAI), counterfactual (CF) explanations interpret a model's decision by suggesting the changes to the input that would lead to a more favourable outcome. To be useful in practice, such an explanation should change few features and change them as little as possible, properties known as sparsity and proximity. However, for numerical features, existing methods typically encourage small changes rather than leaving them exactly unchanged, whether they are model-agnostic and amortised, or gradient-based with full access to the model. In this paper, we propose FlowCF, a model-agnostic generative method that frames CF generation as *sparse transport* from the factual to the target class. We solve this transport with flow matching, which we extend to mixed feature types with a novel mixed flow operator, and exploit the resulting geometry to optimise for exact numerical sparsity through a gating network that leaves selected features exactly unchanged. Extensive experiments on six benchmark datasets demonstrate that FlowCF produces the best numerical sparsity and proximity, changing 16% of the numerical features where the best baseline changes 90%, while reducing numerical displacement by 70% relative to the best baseline and remaining comparable on the other desiderata.

1 INTRODUCTION

Given a model's decision on an instance, a counterfactual (CF) explanation suggests the changes to the input that would lead to a more favourable outcome. This answers a what-if question about that instance, which makes CFs directly actionable, and a popular explanation choice in domains where decisions affect individuals, such as credit, hiring, and medicine, where the data is typically tabular, with both numerical and categorical features Panagiotou et al. (2024).

Useful CFs must satisfy several conflicting properties, known as desiderata Verma et al. (2024). The most common desiderata are: *validity* (the prediction changes), *proximity* (the CF stays close to the factual instance), *sparsity* (few features change), and *actionability* (respect what the user can change), while many more have been proposed, such as *plausibility*, *diversity*, and *amortised inference*, with each method typically targeting a different subset of them. Notably, sparsity and proximity determine how easy a CF is to understand and act on, since explanations are more effective when they involve few features Miller (2019), and smaller changes are more realistic to implement Verma et al. (2024).

Existing methods approach CF explanation generation in different ways and assume different access to the model they explain, but all solve the same underlying task: moving an instance from one class (decision) to a targeted other. Optimisation-based methods such as DiCE Mothilal et al. (2020) and TABCF Panagiotou et al. (2024) search for counterfactuals using gradient descent, in the input space or latent space, which requires access to gradients through the model and one optimisation run per explanation. Generative methods such as CCHVAE Pawelczyk et al. (2020) and DiCoFlex Furman et al. (2026) instead learn the structure of the data with a variational autoencoder or a normalising flow, and can amortise generation into a single forward pass.

In this work, we observe that existing methods fall short on sparsity and proximity, especially for numerical features. We distinguish *exact numerical sparsity*, where features are left exactly unchanged, from *approximate or ϵ-sparsity* Furman et al. (2026), where sufficiently small changes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Desired properties of existing counterfactual methods compared to FlowCF.

	model-agnostic	amortised	actionability/ immutability	exact numerical sparsity	no black-box queries at inference
Wachter	✗	✗	✗	✗	✗
DiCE	✗	✗	✓	✗	✗
TABCF	✗	✗	✗	✗	✗
REVISE	✗	✗	✓	✗	✗
CCHVAE	✓	✗	✓	✗	✗
DiCoFlex	✓	✓	✓	✗	✗
FlowCF	✓	✓	✓	✓	✓

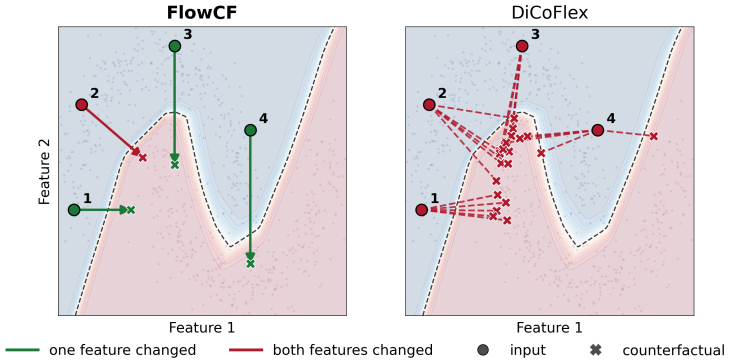

Figure 1: CFs on the two-moons dataset. While DiCoFlex (closest baseline) alters both features, we (FlowCF) achieve exact numerical sparsity by keeping unnecessary features strictly unchanged.

are treated as unchanged. Exact numerical sparsity is difficult to optimise directly, since the L_0 count of changed features is non-differentiable and therefore not directly optimisable by gradient descent Louizos et al. (2017). Existing methods therefore typically encourage sparsity indirectly, for example through L_1 regularisation or post-hoc pruning, rather than incorporating exact numerical sparsity into the generative mechanism itself (see Figure 1). This distinction matters both practically and for evaluation: small non-zero changes still constitute feature changes, increasing the number of interventions required from the user, while tolerance-based sparsity metrics may count them as unchanged depending on the chosen threshold.

To address these limitations, we introduce FlowCF, a generative method that frames CF generation as *sparse transport* from the factual class to the target class. Flow matching is a natural fit for this setting, as it learns a velocity field that moves samples from one continuous distribution to another. Since classical flow matching is defined for continuous data only, we extend it to mixed feature types with a mixed flow operator, following categorical flow maps Roos et al. (2026). The operator takes a single step along a straight path, so that each instance is explained by one displacement vector over its features, and we exploit this geometry to optimise for sparsity. A gating network determines which feature changes are applied, leaving unnecessary numerical features exactly unchanged, while proximity is optimised by minimising the remaining flow vectors. As summarized in Table 1, FlowCF is the only method that is simultaneously model-agnostic and amortised, supports actionable/immutable features, requires no model access at explanation time, and crucially, achieves *exact* numerical sparsity.

Our contributions are: i) identifying the limitation of existing CF methods in achieving exact numerical sparsity; ii) introducing FlowCF, a model-agnostic, amortised method that combines mixed-type flow matching with feature-wise gating to generate exactly sparse and proximal CFs; iii) we demonstrate across six benchmark datasets that FlowCF produces the sparsest and most proximal numerical CFs on every dataset, while remaining comparable on the other desiderata.

Code to reproduce all experiments is available in the supplementary material and at `https://anonymous.4open.science/r/FlowCF`.

2 Related work

Counterfactual explanations have been studied extensively Verma et al. (2024); Pawelczyk et al. (2021). We focus on tabular counterfactual generation and flow-based transport.

Counterfactual explanations for tabular data. Wachter et al. (2017) framed counterfactual search as optimisation for a label flip while remaining close to the factual instance. Later gradient-based methods optimize in input space (DiCE Mothilal et al. (2020)) or generative latent space (REVISE Joshi et al. (2019), TABCF Panagiotou et al. (2024)), requiring per-instance optimization. Model-agnostic alternatives search a conditional VAE latent space (CCHVAE Pawelczyk et al. (2020)) or amortize generation with conditional normalizing flows (DiCoFlex Furman et al. (2026)). Sparsity in these methods is typically encouraged indirectly through regularisation, search, or post-hoc processing rather than enforced in the generative mechanism. In particular, DiCoFlex controls numerical sparsity through its neighbour-selection metric and evaluates it using ϵ-sparsity, where sufficiently small numerical changes are treated as unchanged. In contrast, FlowCF explicitly gates feature-wise transport, leaving selected numerical features exactly unchanged by construction.

Flow-based generation. Normalising flows learn invertible density transformations and have been used to generate plausible and diverse counterfactuals Rezende & Mohamed (2015); Duong et al. (2023); Furman et al. (2026). Flow matching instead learns transport directly by regressing a velocity field along a prescribed interpolant, avoiding invertible architectures and ODE simulation during training Lipman et al. (2022). Extensions handle categorical data on the probability simplex Roos et al. (2026), while rectified flows use straight paths that enable one-step generation Liu et al. (2022). We build on these ideas with a mixed flow-matching operator and explicit gates that enforce exact numerical sparsity while optimising proximity and validity.

3 FlowCF: sparse flow-matching counterfactual explanations

Following prior model-agnostic work Furman et al. (2026), we consider a binary black-box classifier $f : \mathcal{X} \to y \in \{0, 1\}$. We observe only its predicted labels, without access to gradients, model parameters, probabilities, or logits. Each input $x = [x^{\text{num}}, x^{\text{cat}}]$ contains $|N|$ numerical and $|C|$ categorical features. Let π_0 and π_1 denote the empirical distributions of instances assigned to the factual and target classes, respectively. We define the *sparse transport problem* as learning a map $\mathcal{T}$ that moves an instance $x \sim \pi_0$, with $f(x) = 0$, toward π_1, producing a valid counterfactual $x' = \mathcal{T}(x)$ with $f(x') = 1$ while changing as few features as possible.

Flow matching is a natural fit for this problem because it learns a transport between distributions from paired samples. Like other generative counterfactual methods Furman et al. (2026), it amortises generation and avoids a separate optimisation for each counterfactual. However, standard flow matching does not account for CF desiderata like sparsity and proximity, and therefore produces dense displacements.

To address this limitation, we introduce FlowCF. Our method is comprised of two models and is trained in three steps. The first model is a multi-head mixed flow operator $\mathcal{F}_\theta$, designed to support mixed-type data and generate multiple candidate transports toward the target class. The second model is a gate operator G_ψ that determines which feature changes are applied to each instance. At inference, the gates are binarised so that inactive features remain exactly unchanged.

Step 1. Transport (Section 3.1). We train F_θ on pairs of factual and target-class instances, using multiple heads to learn alternative target-directed feature changes.

Step 2. Sparsity (Section 3.2). We freeze F_θ and train G_ψ to remove unnecessary feature changes while maintaining validity.

Step 3. Joint (Section 3.3). We freeze G_ψ and fine-tune F_θ through the learned gates to improve proximity while maintaining validity and staying close to the transport learned in Step 1.

Figure 2 illustrates the effect of each training step on the same four factual instances.

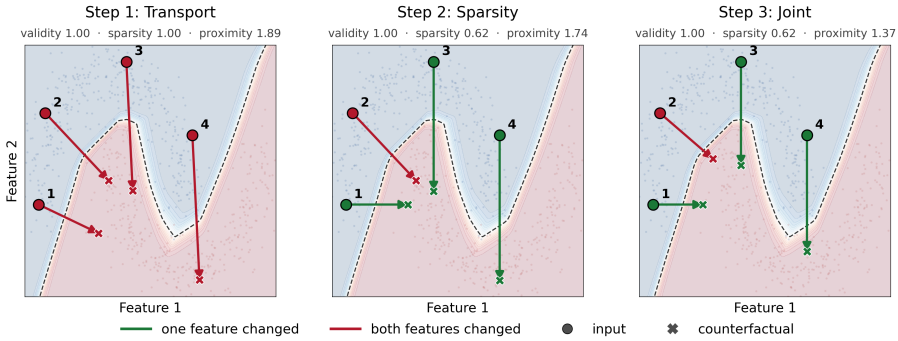

Figure 2: FlowCF's three training steps on samples from the 2D two-moons dataset. Green marks a learned displacement that changed one feature, red both. Step 1 produces valid CFs but changes both features. Step 2 achieves better sparsity making three of the four CFs axis-aligned. Step 3 further improves proximity while preserving validity and sparsity.

3.1 Step 1: Transport

The first step learns a transport from π_0 to π_1. To encourage proximity, following Furman et al. (2026) we pair each factual x_0 with its K nearest neighbours in the target class. This keeps the transport short and ensures that each path ends at an observed target-class instance. However, fitting a single transport to multiple neighbours may collapse distinct target directions into a conditional-mean displacement. We therefore predict H candidate transports and train them *winner-takes-all*, so each pair updates only its best-fitting head.

We learn the transports between these pairs with flow matching Liu et al. (2022); Lipman et al. (2022). For the numerical block, we take the interpolant $x_t^{\text{num}} = (1-t)\,x_0^{\text{num}} + t\,x_1^{\text{num}},\ t \sim \mathcal{U}[0,1]$ and train a velocity field v_θ to predict the displacement from the intermediate point. However, this does not extend to categorical features, since a convex combination of two one-hot vectors is not a valid category. Following the *categorical flow* method Roos et al. (2026), we represent each categorical column in the probability simplex and predict an endpoint distribution μ_θ rather than a velocity. The two parameterisations are equivalent along a straight-line interpolant Roos et al. (2026), while the endpoint form keeps categorical predictions admissible.

Our mixed flow operator $\mathcal{F}_\theta$ supports both output feature types present in the data. It *moves* the numerical block by a displacement, and *replaces* the categorical block with a predicted endpoint, producing one such pair per head $h \in \{1, \dots, H\}$:

$$\mathcal{F}_\theta^h(x,t): \qquad x^{\text{num}} \longmapsto x^{\text{num}} + \underbrace{v_\theta^h(x,t)}_{\text{numerical displacement}}, \qquad x^{\text{cat}} \longmapsto \underbrace{\mu_\theta^h(x,t)}_{\text{categorical endpoint}}, \tag{1}$$

We optimise $\mathcal{F}_\theta$ using the following loss:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\Big[\min_h \Big(\big\|v_\theta^h(x_t,t) - (x_1^{\text{num}} - x_0^{\text{num}})\big\|_2^2 + \text{CE}\big(\mu_\theta^h(x_t,t), x_1^{\text{cat}}\big)\Big)\Big] + \lambda_{\text{val}}\,\mathcal{L}_{\text{val}}\big(\mathcal{F}_\theta(x)\big),$$

$$\text{where} \quad \mathcal{L}_{\text{val}}\big(\mathcal{F}_\theta(x)\big) = \frac{1}{H}\sum_{h=1}^{H} \mathbb{E}\Big[\text{hinge_loss}\big(S(\mathcal{F}_\theta^h(x)), y=1\big)\Big]. \tag{2}$$

The first term learns the numerical and categorical mappings. The numerical velocity is trained by squared error, while CE is the cross-entropy between the predicted and paired target categories, summed over categorical columns using the mean-field factorisation of Roos et al. (2026). The minimum over h implements *winner-takes-all* training, where each pair updates only its best-fitting head, avoiding conditional-mean regression. Validity is averaged over heads so every candidate is pushed across the decision boundary. The second term provides a direct validity signal. Since f is a black-box classifier, we train a surrogate S to reproduce its predictions on $\mathcal{D}$ and freeze it. We then define $\mathcal{L}_{\text{val}}$ using the hinge loss of Mothilal et al. (2020), with λ_{val} controlling its contribution. This loss stops penalising a CF once it reaches the target margin, avoiding unnecessary movement into the target class. We generate using a single Euler step from $t = 0$, i.e. a one-step map $\mathcal{F}_\theta(x) := \mathcal{F}_\theta(x, 0)$.

We find this sufficient for tabular data, and consistent with prior work showing that rectified flows can be distilled into accurate one-step models even in high-dimensional domains Liu et al. (2022).

3.2 STEP 2: SPARSITY

The learned flow of Step 1 changes every feature, without taking the *sparsity* desideratum into account. To address this, we freeze the transport $\mathcal{F}_\theta$ and introduce a gating network, trained to zero out as many feature changes as possible while preserving validity. What we want is a binary mask over features, selecting the ones the flow may change, but such a mask cannot be directly learned by gradient descent, as neither the mask nor the L_0 count is differentiable. Following L_0-style gating Louizos et al. (2017); Cenacchi (2026), we relax it to a continuous gate g, produced by a network a_ψ with one output per feature. Since different candidate transports need different features left open, the gate is conditioned on the displacement $\delta_h = \mathcal{F}^h_\theta(x) - x$ that head h proposes, in addition to the factual. The network itself is shared across heads, and all H gates are produced in one batched evaluation. Our counterfactual operator $\mathcal{T}^h_{\theta,\psi}$ then applies each candidate flow to each feature in proportion to its gate:

$$\mathcal{T}^h_{\theta,\psi}(x) \;=\; \underbrace{(1-g_h)\odot x}_{\text{keep the factual}} \;+\; \underbrace{g_h \odot \mathcal{F}^h_\theta(x)}_{\text{apply the flow}}, \qquad g_h \;=\; \mathcal{G}_\psi(x,\delta_h) \;=\; \sigma\big(a_\psi(x,\delta_h)/T\big), \tag{3}$$

with σ the logistic sigmoid and T a temperature controlling the hardness of the relaxation. We keep the gate deterministic, so that training and inference remain consistent. A gate of 0 keeps a feature at exactly its factual value and a gate of 1 applies the flow in full, while intermediate values scale the displacement of a numerical feature or interpolate a categorical one towards the predicted endpoint. Our gate also naturally guarantees the *actionability* desideratum: setting $g_{h,c} \equiv 0$ for immutable features strictly enforces the constraint, rather than soft-penalizing it, forcing the transport to reach the target class through actionable features alone. These exact-sparsity and actionability guarantees are formalised in Proposition 2 of Appendix B.

We train the gate $\mathcal{G}_\psi$ with three terms, one per desideratum, optimising the composed map $\mathcal{T}_{\theta,\psi}$ of Equation 3 with the transport $\mathcal{F}_\theta$ frozen. For readability, the loss terms that follow suppress the head index; each is averaged over the H candidates.

$$\mathcal{L}_{\text{gate}} \;=\; \underbrace{\mathcal{L}_{\text{val}}(\mathcal{T}_{\theta,\psi}(x))}_{\text{validity}} + \lambda_{\text{sparse}} \underbrace{\mathcal{L}_{\text{sparse}}}_{\text{sparsity}} + \lambda_{\text{prox}} \underbrace{\mathcal{L}_{\text{prox}}}_{\text{proximity}}, \tag{4}$$

$$\text{where} \quad \mathcal{L}_{\text{sparse}} \;=\; \mathbb{E}\Big[\sum_{c\in N\cup C} g_c\Big], \qquad \mathcal{L}_{\text{prox}} \;=\; \mathbb{E}\Big[\sum_{c\in N} |g_c\, v^c_\theta| \;+\; \sum_{c\in C} g_c\, p_c\Big]. \tag{5}$$

We use the same validity term as in Step 1, now evaluated on the *gated* counterfactual rather than on the raw endpoint of the transport. Gating a feature the label flip depends on is therefore penalised; without this signal, the gate would optimise sparsity and proximity alone, and simply keep the features that change least. The sparsity term is the differentiable surrogate for the L_0 count of the sparsity metric in Section 4.4, charging for every feature the gate leaves open. Its weight λ_{sparse} controls how sparse the edit is, closing more features as it increases, until validity begins to suffer. The proximity term charges the scale of the displacement identically for both feature types, using the flow's displacement v^c_θ for numerical features, and for categorical ones the probability mass p_c that the predicted endpoint places outside the original category.

3.3 STEP 3: JOINT

In the final step we unfreeze the transport $\mathcal{F}_\theta$ and freeze the gate $\mathcal{G}_\psi$, optimising the composed map $\mathcal{T}_{\theta,\psi}$ of Equation 3 against the same validity and proximity terms. The flow is therefore re-fit *through* the gate, so that only the features the gate left open carry the label flip. To avoid unstable training and heads from collapsing onto one another, a third term $\mathcal{L}_{\text{drift}}$ keeps $\mathcal{F}_\theta$ close to its Step 1 solution $\mathcal{F}_{\theta'}$.

$$\mathcal{L}_{\text{joint}} \;=\; \mathcal{L}_{\text{val}}(\mathcal{T}_{\theta,\psi}(x)) \;+\; \lambda_{\text{prox}}\,\mathcal{L}_{\text{prox}} \;+\; \lambda_{\text{drift}}\,\mathcal{L}_{\text{drift}}, \qquad \mathcal{L}_{\text{drift}} = \mathbb{E}\big[\,\|\mathcal{F}_\theta(x) - \mathcal{F}_{\theta'}(x)\|_2^2\,\big]. \tag{6}$$

This joint training step improves proximity by shortening the displacement within the features the gate kept open, while preserving sparsity and validity. We visualise this effect in Figure 2.

Inference. A single evaluation of $\mathcal{F}_\theta$ and the binarised gate $\mathcal{G}_\psi$ yields H candidate counterfactuals $\mathcal{T}^h_{\theta,\psi}(x)$. Open categorical features take the argmax of their predicted endpoint, while closed features retain their factual value. We return the candidate ranked highest by our frozen surrogate model. Unlike DiCoFlex Furman et al. (2026), this selection requires no queries to the black-box model.

4 EXPERIMENTAL EVALUATION

This section presents the setup, datasets, baselines, and metrics of our quantitative evaluation.

4.1 EXPERIMENTAL SETUP

Following the literature on CF evaluation Panagiotou et al. (2024); Pawelczyk et al. (2021), all methods run through a single evaluation pipeline, so that factual selection, black-box training and metric computation are identical across methods. For each dataset we train one shared black box, a single 1000-hidden-layer MLP. Every method explains the same $n = 1000$ previously unseen test instances assigned to the undesired class. We run each experiment with three random seeds and report the mean and standard deviation. A valid CF is not guaranteed for every instance Panagiotou et al. (2024), and every metric other than validity is computed over the valid counterfactuals only.

For FlowCF we tune the hyperparameters manually and use the resulting configuration for all datasets, $\{d_S = 256 \times 4,\ d_F = 128 \times 3,\ K = 10,\ H = 8,\ \lambda_{\text{val}} = 2.0,\ \lambda_{\text{sparse}} = 0.3,\ \lambda_{\text{prox}} = 0.375,\ \lambda_{\text{drift}} = 0.01,\ T = 0.2\}$, where d_S and d_F denote the width and depth of the surrogate and flow networks. All experiments use a 16 GB RTX 3080 Ti GPU. FlowCF completes training and inference in under 35 seconds per dataset, substantially faster than several optimisation-based baselines.

4.2 DATASETS

We evaluate on six binary classification datasets, listed in Table 2. Four are mixed-type: Adult, Bank marketing, Credit default and Lending Club Frank (2010); Jagtiani & Lemieux (2019), all standard in the tabular counterfactual literature Panagiotou et al. (2024); Pawelczyk et al. (2021). Two datasets are continuous-only: MAGIC Frank (2010) and HTRU2 Lyon et al. (2016), with only floating-point features. They are deliberately included to showcase the importance of the exact numerical sparsity FlowCF achieves. Although our method is focused on achieving numerical sparsity and proximity, Appendix A shows that FlowCF also performs strongly on an all-categorical dataset.

Table 2: Dataset characteristics.

Dataset	**Training size**	**#Features** $(\mathbf{Num}/\mathbf{Cat})$	**Classification task**
Adult	32,561	4 / 8	Income above or below $50K
Bank marketing	20,002	7 / 9	Subscription to a term deposit
Credit default	27,000	14 / 9	Default on the next payment
Lending Club	50,000	8 / 4	Loan fully paid or charged off
MAGIC	9,510	10 / 0	Gamma versus hadron air shower
HTRU2	8,949	8 / 0	Pulsar versus spurious candidate

We further perform a qualitative visual experiment by encoding the CelebA-HQ Karras et al. (2017) dataset, using VecGAN++ Dalva et al. (2023), which factorises the latent space into six continuous, interpretable feature axes (`bangs`, `glasses`, `hair`, `young`, `male`, `smile`). A counterfactual image is a decoded displacement along these axes. For evaluation, a binary image-space black-box classifier scores the celebrity images for a target attribute (e.g. `male` or `young`).

4.3 BASELINES

We compare against six methods, using CARLA reference implementations Pawelczyk et al. (2021) and official author implementations, evaluated against the same black box and factual instances.

Wachter Wachter et al. (2017) minimizes distance to the original instance subject to a label flip via input-space gradient descent, perturbing numerical features only. **DiCE** Mothilal et al. (2020) optimizes in input space with a regularization term to normalize categorical one-hot blocks and project categories via arg-max, using a post-hoc pass to restore unnecessary numerical edits. **REVISE** Joshi et al. (2019) optimizes the same objective in a VAE latent space, decoding at each step to measure input-space distance. **CCHVAE** Pawelczyk et al. (2020) searches the latent space of a conditional VAE with a growing-sphere procedure until a decoded sample flips the model, avoiding gradients but requiring many queries. **TABCF** Panagiotou et al. (2024) optimizes in the latent space of a transformer-based VAE using a Gumbel-Softmax detokenizer to reconstruct categorical features without post-hoc rounding. **DiCoFlex** Furman et al. (2026) is the closest baseline to our method, using a conditional normalizing flow to generate multiple candidates in one forward pass (amortized), selecting CFs from up to 100 generated candidates per sample based on validity. While DiCoFlex targets sparsity through low-p neighbour selection, it does so only softly via numerical ϵ-sparsity. In contrast, FlowCF achieves exact numerical sparsity: our gates freeze selected features during generation, keeping them identical to their factual values.

4.4 METRICS

We evaluate along the desiderata of Section 1, and in line with literature Pawelczyk et al. (2021), computing every metric directly in the input space so that all methods are scored identically.

Validity. The share of explained instances whose counterfactual flips the black box:

$$\text{Validity } (\uparrow) \;=\; \frac{n_{\text{val}}}{n} \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\big(f(x_i') = 1\big).$$

Sparsity. The fraction of features changed, reported separately per feature type:

$$\text{Sparsity Cat } (\downarrow) = \frac{1}{n_{\text{val}}}\sum_{i=1}^{n_{\text{val}}} \frac{\|x_0^{\text{cat}} - x'^{\,\text{cat}}\|_0}{|C|}, \quad \text{Sparsity Num } (\downarrow) = \frac{1}{n_{\text{val}}}\sum_{i=1}^{n_{\text{val}}} \frac{\|x_0^{\text{num}} - x'^{\,\text{num}}\|_0}{|N|}.$$

For numerical features, we measure exact sparsity: a feature is counted as unchanged only if its value remains exactly unchanged, rather than falling within an ϵ-tolerance.

Proximity. The distance between the counterfactual and the factual instance, measuring how close the explanation remains to the original input rather than how many features it modifies. We standard-normalize the numerical features and average the absolute displacement over $|N|$:

$$\text{Prox Num } (\downarrow) \;=\; \frac{1}{n_{\text{val}}}\sum_{i=1}^{n_{\text{val}}} \frac{\big\|x_0^{\text{num}} - x'^{\,\text{num}}\big\|_1}{|N|}.$$

Here, $\|\cdot\|_0$ and $\|\cdot\|_1$ denote the L_0 (count) and L_1 distance, respectively.

5 RESULTS

This section presents our quantitative results on all tabular datasets (Section 5.1), followed by our visual qualitative results on CelebA-HQ (Section 5.2).

5.1 RESULTS FOR ALL BASELINES ON ALL TABULAR DATASETS

We report the results averaged over all datasets in Table 3, and the per-dataset breakdown in Table 4.

On average, FlowCF achieves the best numerical sparsity and the best numerical proximity of all methods: we change 16% of the numerical features where the best baseline changes 90%, and we change them by 0.229 against 0.767, a 70% smaller displacement. FlowCF is the only method that achieves substantial exact numerical sparsity at all: every baseline changes between 90% and 100% of the numerical features on average. DiCoFlex is the only competitor that targets numerical sparsity directly, but does so softly, encouraging small changes rather than freezing features, as we do with

Table 3: Results averaged over all datasets. **Top 2** counts best or runner-up finishes in Table 4.

	Validity (%) ↑	**Sparsity** (%) ↓ Cat	Num	**Proximity** ↓ Num	**Top 2** ↑
Wachter	0.852±.13	-	1.000±.00	1.113±.05	0
DiCE	**1.000**±.00	**0.141**±.00	0.897±.01	0.981±.04	15
REVISE	0.715±.02	0.170±.00	0.990±.00	0.821±.01	6
CCHVAE	0.857±.10	0.193±.01	1.000±.00	0.767±.02	5
TABCF	0.970±.03	0.400±.08	0.927±.04	1.045±.28	4
DiCoFlex	**1.000**±.00	0.414±.01	0.950±.00	0.814±.02	6
FlowCF (us)	0.970±.02	0.417±.04	**0.159**±.04	**0.229**±.04	**16**

our gating mechanism. Although we do not outperform on validity and categorical sparsity, FlowCF remains competitive on both.

From the per-dataset results of Table 4 we observe that FlowCF is first on numerical sparsity and numerical proximity on all six datasets, with no exceptions. The effect is clearest on the two continuous-only datasets, where every competitor changes every numerical feature of every instance (Sparsity Num $= 100\%$), while FlowCF changes only 16% on MAGIC and 33% on HTRU2. Appendix B formalises this limitation, as a generator that samples from a continuous density, such as DiCoFlex, cannot leave a numerical feature exactly unchanged. Validity stays above 95% on five of the six datasets and never falls below 92%. Categorical sparsity is on average comparable to the other amortised and latent-space methods, TABCF and DiCoFlex.

Table 4: Detailed results on each dataset. **Bold** is best, underlined second best. Categorical sparsity does not apply to the continuous-only datasets MAGIC and HTRU2.

	Validity (%) ↑	**Sparsity** (%) ↓ Cat	Num	**Proximity** ↓ Num	**Validity** (%) ↑	**Sparsity** (%) ↓ Cat	Num	**Proximity** ↓ Num
	Adult				Bank marketing			
Wachter	0.894±.10	-	1.000±.00	0.803±.02	0.765±.21	-	1.000±.00	1.908±.04
DiCE	**1.000**±.00	**0.012**±.00	0.767±.01	2.600±.04	**1.000**±.00	**0.150**±.01	0.856±.02	1.532±.13
REVISE	0.890±.01	0.197±.00	0.994±.00	0.985±.01	0.408±.04	0.170±.01	0.949±.01	1.027±.01
CCHVAE	0.852±.08	0.212±.01	1.000±.00	0.669±.02	0.905±.09	0.213±.00	1.000±.00	0.579±.02
TABCF	0.990±.01	0.294±.11	0.705±.18	1.732±.47	**1.000**±.00	0.265±.10	0.915±.05	1.709±.89
DiCoFlex	**1.000**±.00	0.461±.00	0.872±.01	0.926±.02	**1.000**±.00	0.320±.00	0.898±.01	1.086±.03
FlowCF	0.993±.00	0.507±.05	**0.070**±.03	**0.059**±.02	0.976±.01	0.396±.02	**0.094**±.03	**0.074**±.03
	Credit default				Lending Club			
Wachter	0.774±.14	-	1.000±.00	1.560±.08	0.877±.05	-	1.000±.00	0.772±.04
DiCE	**1.000**±.00	**0.175**±.00	0.902±.00	0.111±.00	**1.000**±.00	0.229±.01	0.860±.00	0.561±.06
REVISE	0.250±.02	0.224±.00	1.000±.00	0.223±.00	0.744±.07	**0.088**±.00	1.000±.00	0.628±.01
CCHVAE	0.957±.04	0.232±.01	1.000±.00	0.503±.01	0.780±.10	0.116±.01	1.000±.00	0.500±.01
TABCF	**1.000**±.00	0.487±.02	0.997±.00	0.476±.20	**1.000**±.00	0.556±.07	0.945±.03	0.758±.06
DiCoFlex	**1.000**±.00	0.455±.01	0.970±.00	0.607±.04	**1.000**±.00	0.421±.01	0.957±.00	0.875±.01
FlowCF	0.924±.05	0.331±.03	**0.095**±.03	**0.043**±.03	0.953±.03	0.433±.03	**0.208**±.06	**0.253**±.15
	MAGIC				HTRU2			
Wachter	0.886±.16	-	1.000±.00	0.652±.08	0.919±.10	-	1.000±.00	0.983±.04
DiCE	**1.000**±.00		1.000±.00	0.369±.01	**1.000**±.00		1.000±.00	0.710±.03
REVISE	**1.000**±.00		1.000±.00	0.769±.01	0.998±.00		1.000±.00	1.293±.02
CCHVAE	0.690±.25		1.000±.00	0.912±.05	0.960±.03		1.000±.00	1.440±.01
TABCF	0.981±.03		1.000±.00	0.881±.01	0.851±.17		1.000±.00	0.715±.06
DiCoFlex	**1.000**±.00		1.000±.00	0.602±.01	**1.000**±.00		1.000±.00	0.789±.02
FlowCF	0.988±.01		**0.158**±.01	**0.327**±.01	0.983±.02		**0.326**±.09	**0.616**±.02

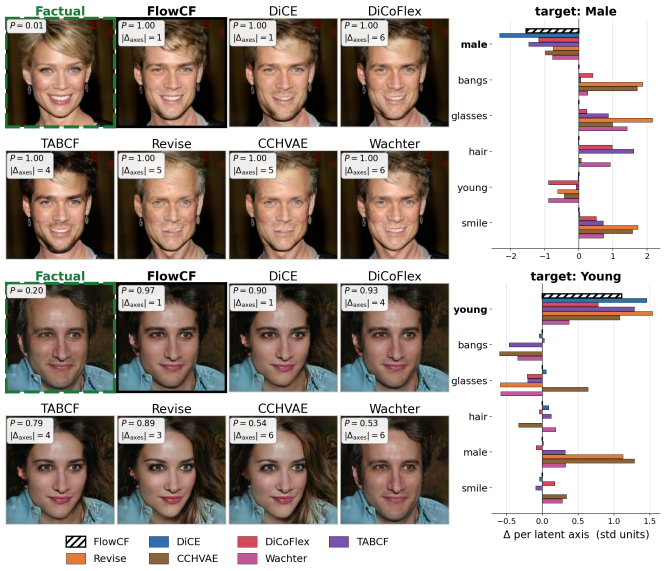

Figure 3: Counterfactuals for two CelebA-HQ instances targeting `male` and `young`. Images show the target-class probability P and number of changed axes $|\Delta_{\text{axes}}|$; bars show the per-axis displacement in standard deviations, with FlowCF hatched.

5.2 Visual qualitative results on CelebA-HQ

Figure 3 shows counterfactuals for two CelebA-HQ instances, targeting the `male` and `young` attributes. As described in Section 4.2, all methods operate on six interpretable VecGAN++ latent axes Dalva et al. (2023), rather than directly in image space. Each image is annotated with the target-class probability and number of changed axes, while the bar plots show the corresponding latent displacements.

All methods produce valid counterfactuals. FlowCF changes only the axis corresponding to the target attribute, without prior knowledge of which axis the classifier uses. DiCE is equally sparse but makes a larger displacement, resulting in a less faithful image. The remaining methods also change attributes correlated with the target. For the `male` target, TABCF darkens the hair, while REVISE and CCHVAE make the subject appear older. For the `young` target, several methods strongly move the `male` axis towards female. In contrast, FlowCF produces targeted and proximal edits while better preserving the other visual attributes of the factual image.

6 Conclusion and future work

This paper presents FlowCF, a method that frames counterfactual generation as a sparse transport map for mixed-type data. We solve the problem using flow matching between undesired and desired distributions, which gives a straight displacement vector towards the target class. We use this geometry to optimise for the sparsity and proximity desiderata: a gate restricts displacement to a few features per instance, and the flow is tuned through the gate to improve proximity. Our method FlowCF is model agnostic, and across six tabular datasets and six competing methods we produce the sparsest and most proximal numerical edits in one forward pass without querying the black box, and remain comparable on the other desiderata to competitors that assume far greater access (e.g. gradients, logits, etc.) to the model they explain.

In future work we want to address further CF desiderata, in particular group and global counterfactuals Ley et al. (2023); Rawal & Lakkaraju (2020), which explain a set of instances rather than a single one. Our transport suits this well: changing the pairing of Step 1 from one-to-many to many-to-many would learn the flow at the group level, and the resulting displacement vectors would be shared across instances by construction. A second direction is interpretability: distilling the learned map into an explainable policy De Toni et al. (2023) would state not only what to change but why.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, to draft portions of the manuscript, to support literature retrieval and discovery, and to assist with research execution. Under research execution we include generating and debugging code for running the experiments, and for aggregating and plotting the results. The research question, the design of the method, and the interpretation of the results are the authors' own. We did not use these tools to generate synthetic data, to produce results that are not backed by executed code, or for any other purpose.

ETHICS STATEMENT

Counterfactual explanations are intended to give individuals actionable insight against automated AI decisions. Our evaluation uses public benchmark datasets that contain sensitive attributes, including Adult, Bank marketing, Credit default and Lending Club. The main risk is a counterfactual that recommends changing an immutable or protected attribute, which is not considered as a valid action. FlowCF addresses this directly, as the gate enforces exact feature actionability/immutability. No new data was collected and no human subjects were involved.

REPRODUCIBILITY STATEMENT

The method, its three training steps and the inference procedure are specified in Section 3. The full hyperparameter configuration, which is identical for all datasets, is given in Section 4, together with the hardware used and the training and inference cost. Datasets and their preprocessing are described in Section 4.2 and Table 2, the baselines and their implementations in Section 4.3, and the evaluation metrics in Section 4.4. All methods are run through a single evaluation pipeline with a shared black box and identical factual instances, and every reported number is the mean over three seeds, with standard deviations in all result Tables. The anonymous supplementary material contains the code, the configuration files, and the instructions and environment specification needed to reproduce both FlowCF and all the baselines. The same code is also available at `https://anonymous.4open.science/r/FlowCF` for easier access. The code will be released through a public repository upon acceptance.

REFERENCES

Filippo Cenacchi. Sparse-by-design cross-modality prediction: L0-gated representations for reliable and efficient learning. *arXiv preprint arXiv:2603.26801*, 2026.

Yusuf Dalva, Hamza Pehlivan, Oyku Irmak Hatipoglu, Cansu Moran, and Aysegul Dundar. Image-to-image translation with disentangled latent vectors for face editing. *IEEE transactions on pattern analysis and machine intelligence*, 45(12):14777–14788, 2023.

Giovanni De Toni, Bruno Lepri, and Andrea Passerini. Synthesizing explainable counterfactual policies for algorithmic recourse with program synthesis. *Machine Learning*, 112(4):1389–1409, 2023.

Tri Dung Duong, Qian Li, and Guandong Xu. Ceflow: A robust and efficient counterfactual explanation framework for tabular data using normalizing flows. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 133–144. Springer, 2023.

Andrew Frank. Uci machine learning repository. *http://archive. ics. uci. edu/ml*, 2010.

Oleksii Furman, Ulvi Movsum-zada, Patryk Marszałek, Maciej Zieba, and Marek Śmieja. Dicoflex: Model-agnostic diverse counterfactuals with flexible control. *Advances in Neural Information Processing Systems*, 38:71592–71623, 2026.

Julapa Jagtiani and Catharine Lemieux. The roles of alternative data and machine learning in fintech lending: Evidence from the lendingclub consumer platform. *Financial management*, 48(4): 1009–1029, 2019.

Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. arxiv. *arXiv preprint arXiv:1907.09615*, 2019.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Dan Ley, Saumitra Mishra, and Daniele Magazzeni. Globe-ce: A translation based approach for global counterfactual explanations. In *International conference on machine learning*, pp. 19315–19342. PMLR, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Robert J Lyon, Ben W Stappers, Sally Cooper, John Martin Brooke, and Joshua D Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 2016.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 607–617, 2020.

Emmanouil Panagiotou, Manuel Heurich, Tim Landgraf, and Eirini Ntoutsi. Tabcf: Counterfactual explanations for tabular data using a transformer-based vae. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pp. 274–282, 2024.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of the web conference 2020*, pp. 3126–3132, 2020.

Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*, 2021.

Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems*, 33:12187–12198, 2020.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, Ismail Ilkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. *arXiv preprint arXiv:2602.12233*, 2026.

Paul Van der Laan. The 2001 census in the netherlands: Integration of registers and surveys. In *CONFERENCE AT THE CATHIE MARSH CENTRE.*, pp. 1–24, 2001.

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56(12):1–42, 2024.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

A Performance on an all-categorical dataset

Although FlowCF targets numerical sparsity and proximity, we also evaluate it in a purely categorical setting. The Dutch census dataset (Van der Laan, 2001) contains $30k$ census records with eleven categorical features and no numerical features.

As shown in Table 5, FlowCF achieves 0.995 validity while changing only 17.3% of the features, compared with 39.0% for DiCoFlex, our closest methodological competitor. Its validity is comparable to DiCE, despite FlowCF being model-agnostic and amortised rather than requiring gradient-based optimisation for each instance.

Table 5: Results on Dutch census, an all-categorical dataset.

	Validity ↑	**Sparsity Cat ↓**
Wachter	–	–
DiCE	<u>0.999</u>$\pm.00$	**0.120**$\pm.00$
REVISE	0.004$\pm.00$	0.161$\pm.02$
CCHVAE	0.972$\pm.03$	<u>0.158</u>$\pm.00$
TABCF	0.941$\pm.07$	0.342$\pm.04$
DiCoFlex	**1.000**$\pm.00$	0.390$\pm.00$
FlowCF	0.995$\pm.01$	0.173$\pm.01$

B Theoretical properties of FlowCF

We use the notation of Section 3. For a head h, $v_h := v_\theta^h(x, 0)$ denotes its numerical displacement in Equation (1), and $\bar{g}_h \in \{0,1\}^{|N|+|C|}$ denotes its binarised inference-time gate.

Proposition 1 (Finite selection from absolutely continuous candidates does not produce exact coordinate preservation)**.** *Consider a generator that, conditional on a factual x, samples finitely many candidates whose numerical blocks are absolutely continuous with respect to Lebesgue measure, and returns one of them unmodified according to any selection rule, without rounding, clipping, projection, immutable-feature restoration, or other post-processing. Then, for every $j \in N$,*

$$\Pr(x'_j = x_j \mid x) = 0.$$

The same holds conditional on any positive-probability event, including validity. Hence, whenever $|N| > 0$, all numerical features change almost surely and $\text{Sparsity Num} = 1$.

Proof. For a single candidate, the event of retaining feature j is the hyperplane $\{z : z_j = x_j\}$, which has Lebesgue measure zero and therefore probability zero under absolute continuity. Since the returned x' is one of the candidates and is not post-processed, it retains feature j only if at least one of the finitely many candidates does, which is a finite union of probability-zero events. Conditioning on an event of positive probability preserves this result. Finally, a union bound over the $|N|$ numerical features shows that all of them change almost surely. □

DiCoFlex Furman et al. (2026) samples candidates using a conditional normalising flow, an invertible transformation of a Gaussian with a nonsingular Jacobian, and selects one of at most 100 candidates. Proposition 1 therefore applies to its raw samples. Consistent with this result, DiCoFlex changes 100% of the numerical features on MAGIC and HTRU2, whose features are all real-valued (Table 4). On the mixed-type datasets, numerical features are integer-valued, so their values lie on a discrete grid rather than admitting a density, and Proposition 1 does not apply to them. This explains why DiCoFlex's Sparsity Num falls below 1 there. Exact numerical sparsity on real-valued features instead requires placing probability mass exactly on the factual value, as achieved by the binarised gate of FlowCF.

Proposition 2 (Gate guarantees)**.** *For any parameters θ, ψ, factual x, and head h, the inference-time counterfactual $x' = \mathcal{T}^h_{\theta,\psi}(x)$ obtained with the binarised gate $\bar{g}_h$ satisfies:*

(i) Exact sparsity and actionability. *If $\bar{g}_{h,c} = 0$, then $x'_c = x_c$. Hence the number of changed features is at most $\|\bar{g}_h\|_0$, and any feature whose gate is fixed to zero is never changed.*

(ii) Gating never increases numerical displacement. *For any relaxed gate* $g_h \in [0,1]^{|N|+|C|}$ *and every* $j \in N$*,*

$$|x'_j - x_j| \leq |v_{h,j}|.$$

Consequently, for every $q \geq 1$*,*

$$\|x'^{\,\mathrm{num}} - x^{\mathrm{num}}\|_q \leq \|v_h\|_q.$$

Proof. On the numerical block, Equation (3) gives

$$x'^{\,\mathrm{num}} - x^{\mathrm{num}} = g_h^{\mathrm{num}} \odot v_h.$$

If $g_{h,j} = 0$, then $x'_j = x_j$; if $g_{h,j} \in [0,1]$, then $|x'_j - x_j| \leq |v_{h,j}|$. This proves (ii) and the numerical part of (i). A closed categorical gate retains the factual category, completing (i). □