CounterFlowNet: From Minimal Changes to Meaningful Counterfactual Explanations

Oleksii Furman[1*] **Patryk Marszałek**[2,3] **Jan Masłowski**[2,3] **Marek Śmieja**[2]
Maciej Zięba[1,4] **Piotr Gaiński**[2,3]

[1]Wrocław University of Science and Technology, Wrocław, Poland
[2]Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
[3]Doctoral School of Exact and Natural Sciences, Jagiellonian University, Kraków, Poland
[4]Tooploox Sp. z o.o., Poland

Abstract

Counterfactual explanations (CFs) provide human-interpretable insights into model's predictions by identifying minimal changes to input features that would alter the model's output. However, existing methods struggle to generate multiple high-quality explanations that (1) affect only a small portion of the features, (2) can be applied to tabular data with heterogeneous features, and (3) are consistent with the user-defined constraints. We propose COUNTERFLOWNET, a generative approach that formulates CF generation as sequential feature modification using conditional Generative Flow Networks (GFLOWNET). COUNTERFLOWNET is trained to sample CFs proportionally to a user-specified reward function that can encode key CF desiderata: validity, sparsity, proximity and plausibility, encouraging high-quality explanations. The sequential formulation yields highly sparse edits, while a unified action space seamlessly supports continuous and categorical features. Moreover, actionability constraints, such as immutability and monotonicity of features, can be enforced at inference time via action masking, without retraining. Experiments on eight datasets under two evaluation protocols demonstrate that COUNTERFLOWNET achieves superior trade-offs between validity, sparsity, plausibility, and diversity with full satisfaction of the given constraints.[1]

1 Introduction

Counterfactual explanations (CFs) have become a central component of explainable artificial intelligence (XAI), offering human-interpretable insight into complex machine learning models [Guidotti, 2024]. They provide actionable guidance of the form: "Your loan was denied; had your income been $5,000 higher, you would have been approved" [Wachter et al., 2017], thereby supporting regulatory objectives such as those outlined in the GDPR [Goodman and Flaxman, 2017] and forming the basis for algorithmic recourse [Karimi et al., 2021, Ustun et al., 2019]. By proposing minimal and semantically meaningful changes to input features that would alter a model's prediction, CFs reveal the underlying factors driving a decision and suggest feasible alternatives to achieve a desired outcome. Their value is particularly pronounced in sensitive tabular domains such as finance, healthcare, and legal decision-making, where understanding both model behavior and plausible counterfactual scenarios is critical [Zhou et al., 2023, Guidotti et al., 2019].

Despite recent progress, existing methods for generating counterfactual explanations still struggle to provide multiple high-quality, actionable examples. First, many methods produce dense perturbations

*Corresponding author: oleksii.furman@pwr.edu.pl
[1]Code is available at https://github.com/ofurman/CounterFlowNet.

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

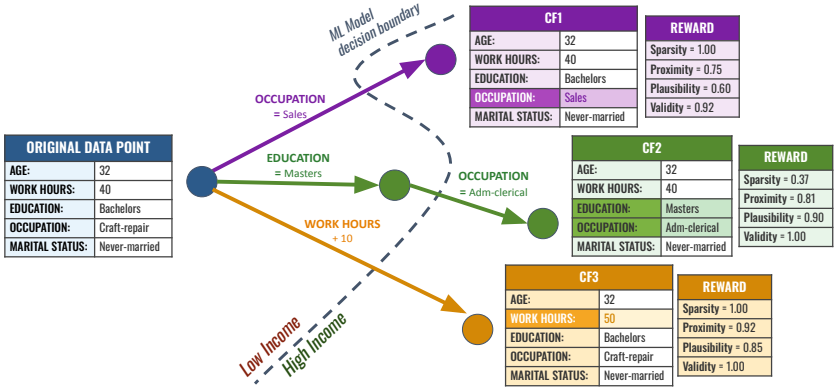


Figure 1: **COUNTERFLOWNET** generates multiple counterfactual explanations by framing feature modification as a sequential decision process. Given an original data point (left), COUNTERFLOWNET samples multiple valid CFs (CF1–CF3) proportional to a composite reward, naturally balancing sparsity, proximity, plausibility, and validity without requiring separate optimization.

that modify numerous features simultaneously, making the resulting explanations difficult to interpret and act upon in practice. Second, optimization-based methods tend to exhibit mode-seeking behavior, converging to a single counterfactual rather than exploring diverse recourse paths [Wielopolski et al., 2024]. Third, generative approaches lack a natural training signal: no distribution of high-quality counterfactuals exists to learn from, so methods resort to heuristic proxies, typically opposite-class nearest neighbors [Pawelczyk et al., 2020], that are not aligned with the counterfactuals we actually want. Finally, enforcing actionability constraints such as feature immutability or monotonicity remains challenging, as most methods rely on model retraining or post-hoc filtering rather than incorporating these constraints natively during inference [Karimi et al., 2021, Ustun et al., 2019].

We propose COUNTERFLOWNET, a novel generative method that addresses these limitations through a fundamentally different perspective. Rather than optimizing a single counterfactual or a surrogate objective, COUNTERFLOWNET uses conditional Generative Flow Networks (GFLOWNET) to sample a diverse set of high-quality CFs guided by a user-defined reward that flexibly encodes key desiderata: validity, proximity, sparsity, and plausibility. At convergence, the sampling distribution is proportional to the reward, so the generative model is directly aligned with the specified desiderata rather than with a proxy. COUNTERFLOWNET constructs explanations through sequential feature modifications, which enforces only necessary changes to the input and naturally handles discrete and continuous attributes in a unified way; actionability constraints such as immutability and monotonicity are enforced directly within the policy's valid action set, removing the need for post-hoc filtering (see Figure 1).

Our contributions can be summarized as follows:

- We introduce a novel generative framework for counterfactual generation, in which CFs are sampled with a probability proportional to the user-defined reward that balances validity, sparsity, proximity and plausibility with user's preference coefficients.
- Our framework supports mixed tabular data through a unified sequential generation process that handles both continuous and discrete attributes via a two-stage state space (feature selection + value assignment) as well as enforces sparsity of modifications.
- COUNTERFLOWNET enforces actionability constraints natively through constraint-aware action masking, enabling immutability and monotonicity without post-hoc filtering or model retraining.
- We perform extensive experimental verification on two well-established benchmarks demonstrating a superior trade-off between diversity, sparsity, and validity compared to existing optimization-based and generative methods.

2 Related Work

Counterfactual Explanations. Wachter et al. [2017] formalized counterfactual explanations as an optimization problem balancing prediction change with proximity to the original input. Mothilal et al. [2020] extended this framework with DiCE, introducing Determinantal Point Process regularization for diversity and one-hot encoding for categorical features. Pursuing plausibility, FACE [Poyiadzi et al., 2020] searches along density-weighted k-NN paths, while REVISE [Joshi et al., 2019] constrains search to VAE latent spaces. For actionability, Ustun et al. [2019] distinguished mutable from immutable features with monotonicity constraints, and Karimi et al. [2021], who established theoretical foundations connecting CFs to causal interventions. However, both diversity and feature heterogeneity remain auxiliary concerns in these optimization-based frameworks. The L2C [Vo et al., 2023] formulates counterfactual generation as a learning problem over counting constraints, specifically designed for discretized data. Other notable approaches include MCCE [Redelmeier et al., 2024], which uses a Monte Carlo method to explore the manifold of valid CFs, and COPA [Bui et al., 2022], which generates CFs for structured policies and has been adapted for tabular data.

Generative Approaches. Generative methods amortize counterfactual generation but can be difficult to align with counterfactual quality metrics. VAE-based approaches such as C-CHVAE [Pawelczyk et al., 2020] learn class-conditional latent representations, while flow-based methods such as CeFlow [Duong et al., 2023] and DiCoFlex [Furman et al., 2025] enable efficient sampling. LAPACE [Jiang et al., 2026] interpolates an instance's latent code toward the target-class centroids of a label-conditional Gaussian-mixture VAE, yielding one CF per mixture component and enforcing differentiable actionability constraints by gradient correction in the latent space. However, VAE and flow-based architectures are primarily designed for continuous spaces, the relationship between sample probability and counterfactual quality depends on the training distribution rather than being explicitly specified, actionability constraints require either architectural modifications or post-hoc enforcement, and many methods require ground-truth counterfactual examples for supervised training, yet such ground truth does not exist in practice.

Generative Flow Networks. GFLOWNET [Bengio et al., 2021] learn stochastic generators that sample objects with probability proportional to a reward, $P_\theta(x) \propto R(x)$, enabling diverse sampling from an explicitly defined quality function. They have been applied to a range of compositional domains, including biological sequences, molecular generation, and Bayesian structure learning [Jain et al., 2022, Gaiński et al., 2025, Deleu et al., 2022]. For explainability, Li et al. [2023] proposed GFlowExplainer for diverse subgraph explanations, highlighting the potential of GFLOWNETs beyond standard generative modeling.

3 Method

COUNTERFLOWNET casts counterfactual generation as a sequential decision process whose terminal states are CFs sampled in proportion to a user-defined reward. Three design choices distinguish our framework: (i) a Markov decision process with a *two-stage action* that selects a feature index and then its new value, naturally accommodating heterogeneous tabular data; (ii) a *composite reward* that combines validity, proximity, sparsity, and plausibility, so that quality is encoded directly in the sampling distribution rather than via a surrogate objective; and (iii) *constraint-aware action masking* that enforces actionability requirements at sampling time without retraining.

After restating the counterfactual desiderata (Section 3.1) and recalling the relevant GFLOWNET background (Section 3.2), we instantiate these three ideas: Section 3.3 formalizes the MDP and the two-stage action, Section 3.4 defines the composite reward, and Sections 3.5 and 3.6 cover training and constrained sampling.

3.1 Counterfactual Explanations Desiderata

Given a classification model h and an initial input example $\mathbf{x}_0$ within a d-dimensional real space $\mathbb{R}^d$, the goal is to extract counterfactual instances. A counterfactual $\mathbf{x}' \in \mathbb{R}^d$ represents a sample, which:

(c1) is classified to the target class y' ($y' \neq h(\mathbf{x}_0)$) by the model h, i.e. $h(\mathbf{x}') = y'$,
(c2) lies close enough to $\mathbf{x}_0$ according to the given distance measure, $d(\mathbf{x}_0, \mathbf{x}')$,
(c3) is plausible (in-distribution sample),

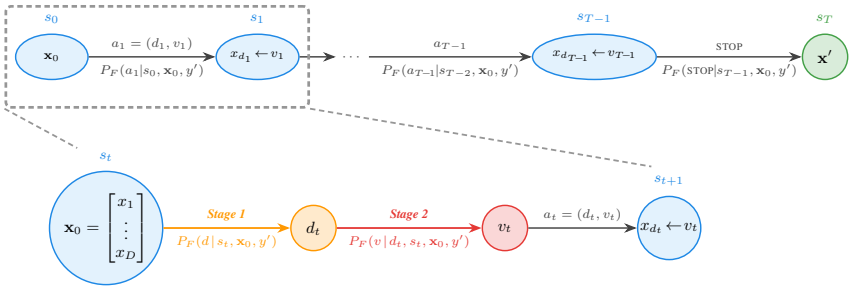


Figure 2: Overview of the COUNTERFLOWNET generative process. **Top:** The forward policy P_F constructs a counterfactual $\mathbf{x}'$ from source instance $\mathbf{x}_0$ through sequential feature modifications until a STOP action terminates the trajectory. **Bottom:** Each action is decomposed into two stages: sampling a feature index $d_t \sim P_F(d \mid s_t, \mathbf{x}_0, y')$ (*Stage 1*, Eq. 7), and sampling its new value $v_t \sim P_F(v \mid d_t, s_t, \mathbf{x}_0, y')$ (*Stage 2*, Eq. 8).

(c4) modifies the minimal number of attributes (sparsity),
(c5) is actionable, it leaves immutable features unchanged and moves monotone features only in the permitted direction [Ustun et al., 2019].

Typically, for many applications, there is no single best counterfactual example for a given data point satisfying the above postulates. Therefore, we are interested in modeling a conditional distribution $P(\mathbf{x}'|\mathbf{x}_0, y')$ of CFs given an example $\mathbf{x}_0$ and the target class y'. By sampling from $P(\mathbf{x}'|\mathbf{x}_0, y')$, we obtain diverse counterfactual options, which can be presented to the user. We adopt GFLOWNETs as the generative backbone because they sample from a distribution defined implicitly by a reward, which is exactly the form we need to encode the desiderata (c1)–(c4) directly into the sampling distribution rather than via a proxy. While (c5) is what makes a CF a recourse the user can act on [Ustun et al., 2019, Karimi et al., 2021]. Rather than penalizing violations in the reward, we remove forbidden actions from the policy via action masking (Section 3.6). This guarantees that every sampled CF obeys the encoded feature-level rules, which can be changed at inference time without retraining.

3.2 Foundations of GFlowNets

A GFLOWNET [Bengio et al., 2021] learns a stochastic forward policy P_F over a directed acyclic graph $G = (\mathcal{S}, \mathbb{A})$ that builds terminal objects $s_f \in \mathcal{S}^f$ via sequences of constructive actions, starting from a unique source state s_0. Given a non-negative reward $R : \mathcal{S}^f \to \mathbb{R}^+$, the policy is trained, e.g. by minimizing the Trajectory Balance objective [Malkin et al., 2022], so that at convergence terminal states are sampled in proportion to the reward:

$$P_F(s_f) = \frac{1}{\mathcal{Z}} R(s_f) \propto R(s_f), \tag{1}$$

where $\mathcal{Z}$ is the normalization constant. This reward-proportional property is what we exploit to align the generative process with counterfactual desiderata. We defer the flow-matching and Trajectory Balance derivations to Appendix A.1.

3.3 Markov Decision Process of COUNTERFLOWNET

The generative process of COUNTERFLOWNET is conditioned on the original input $\mathbf{x}_0$ and the target class y'. We adapt the GFLOWNET notation from Section 3.2 by setting $s_0 = \mathbf{x}_0$, allowing CFs $\mathbf{x}'$ to be the terminal state $\mathbf{x}' \in \mathcal{S}^f$, and conditioning the P_F, P_B and R with $(\mathbf{x}_0, y')$. For transparency, we omit the explicit conditioning on y' if it is clear from context.

The actions for a given state represent possible feature modification decisions $a = (d, v)$, where d represents index of the feature to be modified, $d \in \{1, \ldots, D\}$, and $v \in V(d)$ is the updated feature value from a set of allowed features values for feature d. We can describe the process of creating counterfactual $\mathbf{x}'$ from $\mathbf{x}_0$ with the trajectory:

$$\tau_{\mathbf{x}_0 \to \mathbf{x}'} = (s_0 = \mathbf{x}_0, s_1, \ldots s_{T-1}, s_T = \mathbf{x}'), \tag{2}$$

where s_i represents the state after taking the action $a_i = (d_i, v_i)$ that set the feature d_i with value v_i, see Figure 2.

Two-stage action decomposition. Each action $a_t = (d_t, v_t)$ is decomposed into two sequential decisions, illustrated in the bottom row of Figure 2: in *Stage 1* the policy samples a feature index $d_t \sim P_F(d \mid s_t, \mathbf{x}_0, y')$, and in *Stage 2* it samples the new value $v_t \sim P_F(v \mid d_t, s_t, \mathbf{x}_0, y')$ from the index-conditional distribution over the valid domain $V(d_t)$. This factorization is what allows a single discrete-action policy to handle heterogeneous (categorical and numerical) features uniformly, since each feature's domain $V(d)$ can be specified independently. The explicit forms of the two stage policies are given in Equations (7) and (8).

Similarly to unconditioned GFlowNet (Equation (1)), properly trained P_F of CounterFlowNet is in theory guaranteed to sample x' proportionally to a given reward:

$$P_F(\mathbf{x}'|\mathbf{x}_0, y') \propto R(\mathbf{x}'|\mathbf{x}_0, y'), \tag{3}$$

This formulation allows us to directly encode the notion of a desirable counterfactual into the generative process through the reward function $R(\mathbf{x}'|\mathbf{x}_0, y')$, which assigns higher mass to CFs that better satisfy the desired criteria (c1–c4).

Further in this section, we define our $R(\mathbf{x}'|\mathbf{x}_0, y')$, and describe parameterization and training of P_F.

3.4 Reward function of CounterFlowNet

The reward function $R(\mathbf{x}'|\mathbf{x}_0, y')$ of CounterFlowNet quantifies counterfactual quality by aggregating the core desiderata (c1–c4) into a single scalar signal. We adopt a multiplicative composition of four components:

$$R(\mathbf{x}'|\mathbf{x}_0, y') = R_{\text{v}}(\mathbf{x}'|y')^{\lambda_v} \cdot R_{\text{d}}(\mathbf{x}'|\mathbf{x}_0)^{\lambda_d} \cdot R_{\text{p}}(\mathbf{x}')^{\lambda_p} \cdot R_{\text{s}}(\mathbf{x}'|\mathbf{x}_0)^{\lambda_s}, \tag{4}$$

where R_{v} measures validity, R_{d} penalizes distance (proximity), R_{p} assesses plausibility, and R_{s} encourages sparsity. The exponents $0 \leq \lambda_v, \lambda_d, \lambda_p, \lambda_s \leq 1$ serve dual purposes: they control the relative importance of each criterion and allow users to selectively enable or disable components (setting $\lambda = 0$). The multiplicative structure ensures that *all* criteria must be met, which practically means that if any factor is zero (e.g., an invalid class for validity component $R_{\text{v}}(\mathbf{x}'|y')$), the total reward is 0.

Each component encodes one desideratum from Section 3.1: R_{v} targets validity (c1), R_{d} proximity (c2), R_{p} plausibility (c3), and R_{s} sparsity (c4):

$$\begin{aligned}
R_{\text{v}}(\mathbf{x}'|y') &= \text{clip}\Big(1 - \big(h(y_0|\mathbf{x}') - h(y'|\mathbf{x}')\big) - \varepsilon,\ 0,\ 1\Big), \\
R_{\text{d}}(\mathbf{x}'|\mathbf{x}_0) &= \exp\big(-\|\mathbf{x}' - \mathbf{x}_0\|_p\big), \\
R_{\text{p}}(\mathbf{x}') &= \exp\big(\min(0,\ \text{LOF}(\mathbf{x}') + 1)\big), \\
R_{\text{s}}(\mathbf{x}'|\mathbf{x}_0) &= \exp\Big(-\max\big(\|\mathbb{I}(\mathbf{x}' \neq \mathbf{x}_0)\|_0 - 1,\ 0\big)\Big).
\end{aligned} \tag{5}$$

Here $h(y'|\mathbf{x}')$ is the classifier's probability of the target class, y_0 the original label, $\varepsilon > 0$ a margin controlling the confidence threshold for the class flip, $\|\cdot\|_p$ the L_p norm, LOF$(\cdot)$ the Local Outlier Factor Breunig et al. [2000] measuring local density relative to training neighbors, and $\|\mathbb{I}(\mathbf{x}' \neq \mathbf{x}_0)\|_0$ the count of changed features (the -1 offset gives full sparsity reward for changing at most one feature).[2]

[2]We use a continuous, rather than binary, formulation of R_{v} to provide graduated feedback that guides exploration toward valid CFs; GFlowNet rewards need not be differentiable.

3.5 Training of COUNTERFLOWNET

We train the model using a conditioned form of the Trajectory Balance (TB) objective Malkin et al. [2022] (see Appendix A.1):

$$\mathcal{L}_{TB}(\tau) = \Big(\log F(\mathbf{x}_0, y') + \sum_{t=0}^{T-1} \log P_F(a_t \mid s_t, \mathbf{x}_0, y') - \log R(\mathbf{x}'|\mathbf{x}_0, y') - \sum_{t=0}^{T-1} \log P_B(a_t \mid s_{t+1}, \mathbf{x}_0, y')\Big)^2 \quad (6)$$

where P_F and P_B denote the forward and backward policies, respectively, and $F(\mathbf{x}_0)$ is the flow associated with the initial state $s_0 = \mathbf{x}_0$. Forward policy P_F constructs the counterfactual by iteratively applying perturbations, while backward policy sequentially removes those perturbation, leading to original sample $\mathbf{x}_0$.

Minimizing $\mathcal{L}_{TB}$ over sampled trajectories provably converges P_F to sample CFs $\mathbf{x}'$ proportionally to their reward $R(\mathbf{x}' \mid \mathbf{x}_0, y')$.

Parametrization. We set P_B to be a uniform distribution and do not update it during training. We parametrize $F(\mathbf{x}_0, y')$ as a multi-layer perceptron with ReLU activations that takes as input the one-hot encoding of the original instance $\mathbf{x}_0$ and target label y' and outputs a scalar flow value.

The two-stage forward policy of Section 3.3 is realized as follows. The Stage-1 distribution over feature indices is

$$P_F(d_i \mid s, \mathbf{x}_0, y') = \sigma^n(\mathbf{c})_i, \quad \mathbf{c} = W_1 f(s), \quad (7)$$

where $f : \mathcal{X} \to \mathbb{R}^D$ embeds the state s using a multi-layer perceptron, and $W_1 \in \mathbb{R}^{n \times D}$ is a learnable projection matrix. σ^k is he softmax operator over a k-dimensional logit vector $\mathbf{c} \in \mathbb{R}^k$.

At each step, we apply a mask $\mathcal{M}(s_t, \mathbf{x}_0)$ to the logits, setting $c_j = -\infty$ for indices j that are either immutable by domain constraints or have already been modified in the current trajectory.

The Stage-2 distribution over values $v \in V(d)$, conditioned on the selected index d, is

$$P_F(v_i \mid d, s, \mathbf{x}_0, y') = \sigma^{|V(d)|}(\mathbf{c})_i, \quad c_i = W_d f(s), \quad (8)$$

where $W_d \in \mathbb{R}^{n \times D}$ is the linear embedding layer dependent on chosen feature index d. While GFlowNets are capable of directly parameterizing both continuous and discrete action spaces, we verified that a unified discrete-action formulation works better in our case. To preserve arbitrary low reconstruction error of continuous features (1/128 of a feature original range), we quantize continuous features into $B = 64$ bins and model their values using softmax.

Training Procedure. We train COUNTERFLOWNET using a stochastic off-policy procedure: for each sampled $(\mathbf{x}_0, y')$ pair the forward policy rolls out a trajectory τ under the action mask $\mathcal{M}(s_t, \mathbf{x}_0)$, the composite reward $R(\mathbf{x}' \mid \mathbf{x}_0, y')$ (Equation (4)) is computed at the terminal state, and θ, ψ are updated jointly by gradient descent on $\mathcal{L}_{\text{TB}}$ (Equation (6)). The full procedure is given in Algorithm 1 (Appendix A.2).

3.6 Sampling Counterfactual Explanations

Once trained, the forward policy P_F enables efficient, amortized generation of CFs without iterative optimization. Given an input instance $\mathbf{x}_0$ and target class y', we generate multiple candidates by sampling K independent trajectories from P_F.

Trajectory rollout. Each trajectory τ begins at the initial state $s_0 = \mathbf{x}_0$. At step t, we compute an action mask $m_t = \mathcal{M}(s_t, \mathbf{x}_0)$, then sample a feature index d_t and new value v_t from the masked policy distributions (Equations (7) and (8)). The process terminates when the policy selects the STOP action or a maximum sparsity budget is reached. The final state s_T represents the generated CF $\mathbf{x}'$.

Constraints handling. We enforce actionability (c5) directly during sampling via constraint-aware action masking. The mask m_t removes invalid actions from the forward policy's support, including (i) modifications of immutable features, (ii) actions that would violate monotonicity constraints (e.g., decreasing a feature that must be non-decreasing), and (iii) re-modifying a feature already

Table 1: Results of COUNTERFLOWNET (CFN) and baselines on Protocol A datasets (discretized features). Best in **bold**, second-best underlined.

Method	Spars.(%)↑	Div.(%)↑	H. Mean(%)↑	Val.(%)↑	Cov.(%)↑	Unary(%)↑
German Credit (Logistic Regression)						
L2C	61.35	37.31	46.39	**100.00**	**100.00**	99.06
DICE-R	**88.23**	15.29	26.06	**100.00**	**100.00**	90.81
DICE-G	43.45	37.56	40.29	62.87	90.24	56.66
COPA	57.88	18.88	28.47	44.00	44.00	84.31
MCCE	28.76	33.40	30.91	48.74	**100.00**	58.76
CFN	69.09	**41.26**	**51.65**	99.48	**100.00**	**100.00**
Adult Income (Neural Network)						
L2C	45.70	28.11	34.80	**100.00**	**100.00**	97.62
DICE-R	**89.26**	9.05	16.44	**100.00**	**100.00**	87.15
DICE-G	41.48	26.27	32.14	92.64	**100.00**	72.70
MCCE	24.93	4.58	7.74	30.63	74.76	45.79
CFN	62.36	**38.21**	**47.39**	**100.00**	**100.00**	**100.00**
Graduate Admission (Neural Network)						
L2C	42.23	37.90	39.94	**100.00**	**100.00**	**100.00**
DICE-R	**66.25**	30.93	42.15	**100.00**	**100.00**	85.30
DICE-G	23.05	**47.54**	31.04	92.91	**100.00**	66.69
MCCE	17.39	22.98	19.51	43.79	84.60	79.11
CFN	55.41	46.18	**50.37**	99.53	**100.00**	**100.00**
Student Performance (Logistic Regression)						
L2C	55.32	29.54	38.51	**100.00**	**100.00**	**100.00**
DICE-R	**87.60**	13.64	23.60	**100.00**	**100.00**	98.99
DICE-G	39.20	**39.88**	38.54	84.83	**100.00**	60.77
COPA	50.45	25.28	33.68	67.26	67.26	95.32
MCCE	25.97	24.97	25.46	60.98	93.10	67.70
CFN	71.18	31.44	**43.61**	99.63	**100.00**	**100.00**

changed earlier in the same trajectory. Since sampling is performed from the renormalized masked distributions, every generated trajectory respects the specified constraints without retraining or post-hoc filtering.

Candidate Selection. Since P_F samples proportionally to the reward, the batch of K trajectories naturally covers high-quality modes of the counterfactual distribution. We filter the candidates to retain only valid CFs satisfying $h(\mathbf{x}') = y'$, then select the top-N unique instances ranked by $R(\mathbf{x}' \mid \mathbf{x}_0, y')$.

4 Experiments

Our experiments are organized around the three claims of Section 3: (Q1) Does reward-proportional sampling produce CFs that jointly satisfy validity, sparsity, proximity, and plausibility better than methods optimizing a single proxy? (Q2) Does the discrete two-stage action space remain competitive on mixed tabular data (continuous and categorical)? (Q3) Do the reward weights $\boldsymbol{\lambda}$ give predictable control over the resulting trade-offs? (Q4) Does inference-time action masking enforce actionability constraints without retraining? Section 4.2 answers Q1 and Q2 on two benchmark protocols; Section 4.3 addresses Q2–Q4 on Adult. The full experimental setup is given in Appendix B.

4.1 Experimental Setup

Datasets and Evaluation Protocols. We adopt datasets from two recent counterfactual benchmarks that employ complementary evaluation philosophies. **Protocol A** assumes discrete features for all baseline methods [Vo et al., 2023]. Continuous features are quantized into 4 equal-frequency bins prior to classifier training. **Protocol B** evaluates methods using continuous and discrete features [Furman et al., 2025]. Protocol A covers German Credit, Adult, Graduate Admission, and Student Performance; Protocol B covers Adult, Bank Marketing, Default, GMC, and Lending Club, giving eight distinct datasets (Appendix B.1, Table 4).

Baselines. We compare COUNTERFLOWNET against a comprehensive set of state-of-the-art methods, spanning three major paradigms: optimization-based (DiCE, COPA), sampling-based (MCCE), and

Table 2: Results of COUNTERFLOWNET (CFN) and baselines on Protocol B datasets. Baselines use continuous features. CFN uses $B{=}64$ bins and is scored after decoding.

Method	Val. ↑	Prox.-Cont ↓	Spars.-Cat ↓	ϵ-Spars. ↓	LOF ↓	Div. ↑	Modes ↑
Adult Income (Neural Network)							
DICE	**1.00**	0.45	**0.06**	0.50	**0.15**	0.07	**4.96**
CCHVAE	**1.00**	0.52	0.07	0.97	0.24	0.03	0.56
DiCoFlex	**1.00**	0.85	0.49	0.97	0.31	**0.40**	2.17
LAPACE	**1.00**	0.43	0.40	0.53	0.23	0.26	1.86
CFN	**1.00**	**0.19**	0.27	**0.36**	0.20	0.21	3.69
Bank (Neural Network)							
DICE	**1.00**	0.64	**0.09**	0.67	**0.11**	0.13	3.52
CCHVAE	**1.00**	0.67	0.13	0.97	0.18	0.07	0.07
DiCoFlex	**1.00**	0.89	0.40	0.95	0.16	**0.39**	0.85
LAPACE	0.86	0.63	0.37	0.68	0.12	0.33	1.48
CFN	**1.00**	**0.25**	0.23	**0.38**	0.21	0.21	**8.98**
Default (Neural Network)							
DICE	**1.00**	0.33	**0.06**	0.79	0.23	0.06	2.24
CCHVAE	**1.00**	0.49	0.09	0.97	0.37	0.03	0.17
DiCoFlex	**1.00**	0.64	0.56	0.97	0.39	**0.33**	0.33
LAPACE	0.89	0.37	0.47	0.47	**0.20**	0.18	0.59
CFN	0.96	**0.12**	0.26	**0.19**	0.34	0.14	**4.94**
GMC (Neural Network)							
DICE	**1.00**	0.28	**0.05**	0.70	**0.04**	0.05	3.49
CCHVAE	**1.00**	0.40	0.07	0.94	0.36	0.02	0.42
DiCoFlex	**1.00**	0.79	0.83	0.96	0.53	**0.51**	0.69
LAPACE	0.92	0.34	0.89	0.48	0.17	0.09	0.41
CFN	**1.00**	**0.12**	0.63	**0.25**	0.24	0.18	**4.26**
Lending Club (Neural Network)							
DICE	**1.00**	0.86	**0.20**	0.81	**0.03**	0.14	0.49
CCHVAE	**1.00**	0.61	0.65	0.94	0.16	0.04	0.05
DiCoFlex	**1.00**	1.06	0.76	0.94	0.05	**0.34**	0.10
LAPACE	0.88	0.87	0.76	0.78	**0.03**	0.30	0.16
CFN	0.94	**0.56**	0.41	**0.29**	0.21	0.24	**9.23**

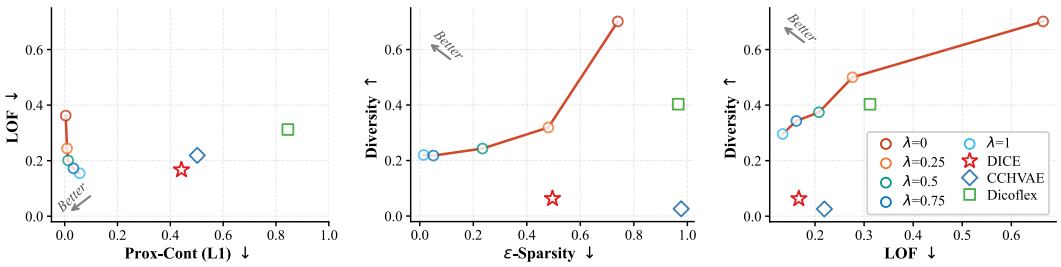
generative/amortized methods (L2C, C-CHVAE, DiCoFlex, LAPACE). A detailed description of each baseline is provided in Appendix B.4.

Evaluation Metrics. We evaluate validity, proximity, sparsity, diversity, plausibility (LOF), and constraint satisfaction, following the metric definitions of Vo et al. [2023] for Protocol A and Furman et al. [2025] for Protocol B. For Protocol B we additionally report **Modes**, the number-of-modes metric of the GFLOWNET literature [Bengio et al., 2021] applied per instance, the mean number of valid, high-reward, and mutually distant CFs offered for one instance. Detailed formulations are in Appendix B.3.

Reward Configuration. Each protocol uses one shared reward configuration across its datasets. **Protocol A**: $\lambda_v{=}1$, $\lambda_s{=}0.01$, $\lambda_d{=}\lambda_p{=}0$, $\varepsilon{=}0.1$. **Protocol B**: $\lambda_v{=}1$, $\lambda_s{=}0.8$, $\lambda_d{=}0.4$, $\lambda_p{=}0.4$, $\varepsilon{=}0.1$, with continuous features quantized into $B{=}64$ bins at generation time. The exception is Lending Club dataset: $\lambda_s{=}0.5$, $\lambda_d{=}0.3$, $\varepsilon{=}0.25$, λ_v, λ_p unchanged. Its classifier has by far the most compressed margins: mean $h(y_0|\mathbf{x}_0) - h(y'|\mathbf{x}_0)$ is 0.60 vs. 0.67–0.82 on the other Protocol B datasets. At $\varepsilon{=}0.1$, an unedited Lending Club instance already earns substantial partial validity credit (mean $R_v{=}0.30$, Eq. 5), so under the multiplicative reward a small edit that stops short of the decision boundary can out-score a genuine class flip through its higher proximity and sparsity terms. Raising ε to 0.25 removes this advantage.

4.2 Main Results

Protocol A (Q1). Table 1 compares methods on the L2C benchmark where all features are discretized. COUNTERFLOWNET demonstrates a superior balance between all key desiderata in this case. As shown in Table 1, our method consistently achieves near-perfect Validity, Coverage, and Unary constraint satisfaction ($\geq$99%), outperforming most baselines, particularly DICE-G, COPA, and MCCE, which often fail to generate valid or constraint-abiding CFs. While DICE-R achieves high sparsity, it does so at the expense of diversity. Conversely, methods like DICE-G and L2C achieve



(a) Varying λ_p (plausibility weight). λ_d=1.0, λ_v=1.0, λ_s=0.4 are fixed

(b) Varying λ_d (proximity weight). λ_p=0.0, λ_v=1.0, λ_s=0.4 are fixed

(c) Varying λ_p (plausibility weight). λ_d=0.0, λ_v=1.0, λ_s=0.4 are fixed

Figure 3: Reward component ablation on the Adult dataset. (a) Increasing plausibility weight λ_p improves LOF at the cost of proximity. (b) Higher proximity weight λ_d yields sparser CFs with lower diversity. (c) Higher plausibility weight λ_p yields more plausible CFs with lower diversity. All COUNTERFLOWNET configurations consistently dominate the baseline Pareto frontiers.

Table 3: Impact of actionability constraints on Adult. CFs quality under three increasingly strict constraint sets, comparing *inference-time* constraint action masking to *retraining* with the same constraints applied during training.

CFN	Val. ↑	Prox.-Cont ↓	Spars.-Cat ↓	ϵ-Spars. ↓	LOF ↓	Div. ↑
Adult Income - Unconstrained						
Retrained	0.99	0.22	0.28	0.41	0.22	0.23
Adult Income - Constraints Set 1						
Retrained	0.99	0.24	0.27	0.45	0.22	0.23
Inference	0.99	0.24	0.27	0.46	0.22	0.23
Adult Income - Constraints Set 2						
Retrained	0.98	0.24	0.28	0.44	0.22	0.21
Inference	0.99	0.25	0.26	0.48	0.23	0.20
Adult Income - Constraints Set 3						
Retrained	0.99	0.33	0.25	0.49	0.24	0.21
Inference	0.97	0.28	0.25	0.50	0.25	0.19

higher diversity but compromise on sparsity. COUNTERFLOWNET achieves the highest Harmonic Mean across all four datasets, demonstrating superior balance between these competing objectives.

Protocol B (Q1, Q2). In the continuous protocol, we compare COUNTERFLOWNET with four state-of-the-art methods adapted to this setting, namely DiCE (KD-Tree), CCHVAE, DiCoFlex, and LAPACE. In the continuous setting (Table 2), COUNTERFLOWNET demonstrates state-of-the-art performance in generating sparse and close CFs while maintaining high diversity. It achieves the best ϵ-sparsity and the best proximity (Prox-Cont) on every dataset, indicating that its modifications are both small in magnitude and few in number. While DiCE and CCHVAE produce slightly sparser changes for categorical features on some datasets, they fall short on numerical sparsity. DiCoFlex achieves the highest diversity, but this spread does not translate into distinct high-quality recourse options. It yields fewer modes per instance, whereas COUNTERFLOWNET ranks first in Modes on four of five datasets. Thus, COUNTERFLOWNET pays for its proximity and sparsity with lower pairwise spread, not with fewer distinct high-quality alternatives. All methods generate valid and plausible (low LOF) CFs, but COUNTERFLOWNET provides an effective trade-off between minimal, interpretable changes and diverse recourse options.

4.3 Ablation Studies

Discretization Granularity (Q2). Binning affects only COUNTERFLOWNET, and decoding moves a feature by at most $V_j/(2B)$, i.e. $V_j/128$ at B=64, well below the ϵ-sparsity tolerance of $0.05\,V_j$, so quantization alone never registers as a change. Varying B on Adult (Appendix C.2) exposes a

trade-off: finer grids lower LOF but worsen proximity and ϵ-sparsity, diversity peaks at coarser grids, and validity stays high throughout, peaking at B=64, the value we use. The bin count shifts the level of the metrics, not the effect of the reward weights above: retraining over $B \in \{16, 64\} \times \lambda_d \times \lambda_p$ (Appendix C.4) gives Spearman rank correlations of 0.75–1.0 between the two bin counts' rankings of the reward settings, so B and $\boldsymbol{\lambda}$ can be chosen independently.

Preference Coefficients (Q3). The composite reward allows practitioners to express their preference over desired counterfactual by adjusting the preference coefficients $\lambda_v, \lambda_d, \lambda_p, \lambda_s$. Figure 3 shows that one can effectively control the properties of generated counterfactuals. Increasing the plausibility weight λ_p improves LOF at the cost of proximity (Fig. 3a) and of diversity (Fig. 3c). Higher proximity weight λ_d produces sparser CFs with reduced diversity (Fig. 3b). Crucially, all COUNTERFLOWNET configurations consistently define a better Pareto frontier than DiCE and C-CHVAE, demonstrating that the framework provides fine-grained control while maintaining great performance. Full quantitative results are in Appendix C.3.

Impact of Constraints (Q4). We evaluate how COUNTERFLOWNET handles user-specified actionability requirements by progressively tightening constraints on the Adult dataset (Table 3). *Set 1* fixes *sex* and *race* and enforces monotonicity on *age*; *Set 2* additionally makes *native country* immutable and enforces monotonicity on *education*; *Set 3* further fixes *marital status*. For each set, we compare two ways of enforcing constraints: (i) *Inference-time* masking, where we apply constraint-aware action masking $\mathcal{M}(s_t, \mathbf{x}_0)$ during sampling without retraining, and (ii) *retraining* with constraints, where the same mask is used throughout training. Across constraint sets, tightening actionability requirements generally degrades proximity and sparsity gracefully, while validity remains high, highlighting a clear trade-off between feasibility constraints and minimality of edits. Overall, inference-time masking achieves performance comparable to retraining while guaranteeing 100% constraint satisfaction by construction, making it a practical way to deploy COUNTERFLOWNET under new user constraints without additional training.

Additional results. Appendix C.1 keeps the MDP and reward of COUNTERFLOWNET fixed and replaces only the generator by random, beam, or evolutionary search, or by a REINFORCE-trained policy: COUNTERFLOWNET offers the most modes per instance on every Protocol B dataset, so its diversity stems from the GFLOWNET objective rather than from the shared action space and reward. Further appendices report runtime (Appendix C.5), feature utilization (Appendix C.6), and qualitative examples (Appendix C.7).

5 Conclusions

We introduced COUNTERFLOWNET, which casts counterfactual generation as sequential feature modification and trains a conditional GFLOWNET to sample CFs in proportion to a user-defined reward over validity, proximity, sparsity, and plausibility. A two-stage action (feature, then value) handles mixed tabular data in a single discrete action space, and action masking enforces immutability and monotonicity at inference time.

The experiments support each design choice. Reward-proportional sampling avoids trading sparsity for diversity: COUNTERFLOWNET attains the highest harmonic mean of sparsity and diversity on all four Protocol A datasets with at least 99% validity, and offers the most distinct high-reward CFs per instance on four of five Protocol B datasets (Q1). Despite operating on a 64-bin grid, it produces the closest and ϵ-sparsest CFs on every Protocol B dataset (Q2). The reward weights shift the trade-offs predictably, with every tested configuration on a better Pareto frontier than the baselines (Q3), and inference-time masking stays within 0.05 of retraining on every metric while satisfying the constraints by construction (Q4). Future work includes richer reward terms such as fairness, modality-specific encoders for text, images, and time series.

Acknowledgments and Disclosure of Funding

Oleksii Furman and Maciej Zięba's work was supported by the National Science Centre (Poland) Grant No. 2024/55/B/ST6/02100.

The research of P. Marszałek and J. Masłowski was supported by the National Science Centre (Poland), grant no. 2023/50/E/ST6/00169.

The work of M. Śmieja and P. Gaiński was supported by the National Science Centre (Poland), grant no. 2022/45/B/ST6/01117.

Some experiments were performed on servers purchased with funds from the flagship project entitled "Artificial Intelligence Computing Center Core Facility" from the DigiWorld Priority Research Area within the Excellence Initiative – Research University program at Jagiellonian University in Krakow. We gratefully acknowledge Poland's high-performance Infrastructure PLGrid ACC Cyfronet AGH for providing computer facilities and support within computational grant no PLG/2025/018842.

We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2026/020022

Competing interests. Maciej Zięba is also affiliated with Tooploox Sp. z o.o. The authors declare no other competing interests.

References

Mohan S Acharya, Asfia Armaan, and Aneeta S Antony. A comparison of regression models for prediction of graduate admissions. In *2019 international conference on computational intelligence in data science (ICCIDS)*, pages 1–5. IEEE, 2019.

Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations. *arXiv preprint arXiv:2009.01884*, 2020.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394, 2021.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Ngoc Bui, Duy Nguyen, and Viet Anh Nguyen. Counterfactual plans under distributional ambiguity. In *International Conference on Learning Representations*, 2022.

Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. In *Proceedings of the 5th Future Business Technology Conference (FUBUTEC 2008)*, pages 5–12, 2008.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022.

Tri 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*, pages 133–144. Springer, 2023.

Oleksii Furman, Ulvi Movsum-zada, Patryk Marszalek, Maciej Zieba, and Marek Smieja. DiCoFlex: Model-agnostic diverse counterfactuals with flexible control. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Piotr Gaiński, Oussama Boussif, Andrei Rekesh, Dmytro Shevchuk, Ali Parviz, Mike Tyers, Robert A. Batey, and Michał Koziarski. Scalable and cost-efficient de novo template-based molecular generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=zssWxiiJZ1`.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, 2017.

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 38(5):2770–2824, 2024.

Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 34(6):14–23, 2019.

Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

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.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with GFlowNets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

Junqi Jiang, Francesco Leofante, Antonio Rago, and Francesca Toni. Synthesising counterfactual explanations via label-conditional gaussian mixture variational autoencoders. In *International Conference on Learning Representations*, volume 2026, pages 82767–82782, 2026.

Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. In *NeurIPS Workshop on Human-Centric Machine Learning*, 2019.

Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 353–362, 2021.

Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

Michał Koziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Chenghao Liu, Mike Tyers, and Robert Batey. RGFN: Synthesizable molecular generation using GFlowNets. *Advances in Neural Information Processing Systems*, 37:46908–46955, 2024.

Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. DAG matters! GFlowNets enhanced explainer for graph neural networks. In *International Conference on Learning Representations*, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. In *Advances in Neural Information Processing Systems*, volume 35, pages 5955–5967, 2022.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

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*, pages 607–617. ACM, 2020.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, pages 3126–3132, 2020.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. FACE: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 344–350, 2020.

Annabelle Redelmeier, Martin Jullum, Kjersti Aas, and Anders Løland. MCCE: monte carlo sampling of valid and realistic counterfactual explanations for tabular data. *Data Min. Knowl. Discov.*, 38(4): 1830–1861, 2024. doi: 10.1007/S10618-024-01017-Y.

Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19, 2019.

Vy Vo, Trung Le, Van Nguyen, He Zhao, Edwin V Bonilla, Gholamreza Haffari, and Dinh Phung. Feature-based learning for diverse and privacy-preserving counterfactual explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2211–2222, 2023.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2): 841–887, 2017.

Patryk Wielopolski, Oleksii Furman, Jerzy Stefanowski, and Maciej Zieba. Probabilistically plausible counterfactual explanations with normalizing flows. In *ECAI 2024 - 27th European Conference on Artificial Intelligence*, volume 392, pages 954–961. IOS Press, 2024. doi: 10.3233/FAIA240584. URL `https://doi.org/10.3233/FAIA240584`.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2): 2473–2480, 2009.

Siqiong Zhou, Upala J Islam, Nicholaus Pfeiffer, Imon Banerjee, Bhavika K Patel, and Ashif S Iquebal. Scgan: Sparse countergan for counterfactual explanations in breast cancer prediction. *IEEE Transactions on Automation Science and Engineering*, 21(3):2264–2275, 2023.

The appendix follows the order of the main paper. Section A complements Section 3 with the GFLOWNET background, the full training algorithm, and candidate selection. Section B details the experimental setup of Section 4: datasets, evaluation protocols, metrics, baselines, and hyperparameters. Section C reports additional results: a generator ablation that isolates the effect of the GFLOWNET objective, the full discretization and reward-weight ablations behind Q2 and Q3, runtime, feature utilization, and qualitative examples. Section D discusses limitations.

A Method Details

A.1 GFlowNet Foundations

We provide here the flow-matching and Trajectory Balance derivations that underlie the proportionality result stated in Equation (1) of the main text. A GFLOWNET operates on a directed acyclic graph (DAG) $G = (\mathcal{S}, \mathbb{A})$ where states $s \in \mathcal{S}$ represent partially constructed objects and actions $a = (s \to s') \in \mathbb{A}$ are constructive transitions, with a unique source state s_0 and a set of terminal states $\mathcal{S}^f$. The reward $R : \mathcal{S}^f \to \mathbb{R}^+$ evaluates terminal nodes.

The goal of training a GFLOWNET is to find a non-negative flow function $F(s \to s')$ satisfying the flow-matching criterion:

$$\forall s : F(s) = \sum_{(s'' \to s) \in \mathbb{A}} F(s'' \to s) = \sum_{(s \to s') \in \mathbb{A}} F(s \to s'), \tag{9}$$

$$\forall_{s_f \in \mathcal{S}^f} : F(s_f) = R(s_f). \tag{10}$$

Satisfying this criterion is equivalent to satisfying the Trajectory Balance objective Malkin et al. [2022] for every trajectory $\tau = (s_0, \ldots, s_T)$:

$$F(s_0) \prod_{t=0}^{T-1} P_F(a_t \mid s_t) \;=\; R(s_f) \prod_{t=0}^{T-1} P_B(a_t \mid s_{t+1}), \tag{11}$$

where P_F and P_B are the forward and backward policies, $P_F(a_t \mid s_t)$ is the probability of taking action a_t at state s_t, P_B governs the deconstruction analogously, and $F(s_0)$ is the flow through the source state. Training under this objective converges to a forward policy P_F that samples terminal states proportionally to the reward as in Equation (1) of the main text.

From proportional sampling to diversity. When training minimizes the TB objective of Equation (11), the forward policy samples terminal states approximately in proportion to their reward. In our conditional setting,

$$P_F(\mathbf{x}' \mid \mathbf{x}_0, y') \approx \frac{R(\mathbf{x}' \mid \mathbf{x}_0, y')}{\mathcal{Z}(\mathbf{x}_0, y')}, \tag{12}$$

where $\mathcal{Z}(\mathbf{x}_0, y') = \sum_{\mathbf{x}' \in \mathcal{X}} R(\mathbf{x}' \mid \mathbf{x}_0, y')$ is the partition function. Summing Equation (12) over a region $\mathcal{X}_A \subset \mathcal{X}$ gives $P_F(\mathcal{X}_A \mid \mathbf{x}_0, y') \approx \sum_{\mathbf{x}' \in \mathcal{X}_A} R(\mathbf{x}' \mid \mathbf{x}_0, y')/\mathcal{Z}(\mathbf{x}_0, y')$, and for two disjoint regions $\mathcal{X}_A, \mathcal{X}_B$ the partition function cancels:

$$\frac{P_F(\mathcal{X}_A \mid \mathbf{x}_0, y')}{P_F(\mathcal{X}_B \mid \mathbf{x}_0, y')} \approx \frac{\sum_{\mathbf{x}' \in \mathcal{X}_A} R(\mathbf{x}' \mid \mathbf{x}_0, y')}{\sum_{\mathbf{x}' \in \mathcal{X}_B} R(\mathbf{x}' \mid \mathbf{x}_0, y')}. \tag{13}$$

A well-trained GFLOWNET therefore spreads probability mass over all high-reward regions rather than concentrating on a single optimum, which is what allows COUNTERFLOWNET to offer several distinct high-quality CFs per instance. With finite training and model capacity the proportionality is only approximate; we measure the resulting diversity empirically with the Modes metric (Section B.3.1) and against generators that share the MDP and reward of COUNTERFLOWNET (Section C.1).

A.2 Training Algorithm

We train COUNTERFLOWNET using a stochastic off-policy procedure outlined in Algorithm 1. Training proceeds by iteratively sampling batches of source instances $\mathbf{x}_0$ and target labels y' from the

Algorithm 1 Training COUNTERFLOWNET with Conditioned Trajectory Balance

Input: Dataset $\mathcal{D}$, Classifier h, Reward weights $\boldsymbol{\lambda}$, Learning rate η
Initialize: Forward policy parameters θ, Conditional flow parameters ψ
Fixed: Backward policy P_B (uniform), Action mask function $\mathcal{M}$
repeat
  Sample batch of pairs $(\mathbf{x}_0, y') \sim \mathcal{D}$
  $\mathcal{L}_{\text{batch}} \leftarrow 0$
  for each $(\mathbf{x}_0, y')$ in batch **do**
    Initialize state $s_0 \leftarrow (\mathbf{x}_0, y')$, trajectory $\tau \leftarrow \emptyset$
    $t \leftarrow 0$
    // Rollout Trajectory
    while s_t is not terminal **do**
      Get valid action mask $m_t \leftarrow \mathcal{M}(s_t, \mathbf{x}_0)$ {Exclude immutable/modified features actions}
      Stage 1: Sample feature index $d_t \sim P_F(d \mid s_t, \mathbf{x}_0, y'; \theta)$ masked by m_t
      Stage 2: Sample value $v_t \sim P_F(v \mid d_t, s_t, \mathbf{x}_0, y'; \theta)$ constrained by domain limits $V(d)$
      Execute action $a_t = (d, v)$ to get s_{t+1}
      $\tau \leftarrow \tau \cup \{(s_t, a_t, s_{t+1})\}$
      $t \leftarrow t + 1$
    end while
    Let $\mathbf{x}' = s_t$ be the generated counterfactual
    // Compute Objectives
    Compute reward $R(\mathbf{x}'|\mathbf{x}_0, y')$ using Eq. 4
    $\log P_F(\tau) \leftarrow \sum_{i=0}^{t-1} \log P_F(a_i|s_i; \theta)$
    $\log P_B(\tau) \leftarrow \sum_{i=0}^{t-1} \log P_B(a_i|s_{i+1})$ {Uniform fixed prob}
    // Trajectory Balance Loss (Eq. 6)
    $\Delta_{TB} \leftarrow \log F_\psi(\mathbf{x}_0, y') + \log P_F(\tau) - \log R(\mathbf{x}'|\mathbf{x}_0, y') - \log P_B(\tau)$
    $\mathcal{L}_{\text{batch}} \leftarrow \mathcal{L}_{\text{batch}} + (\Delta_{TB})^2$
  end for
  // Optimization Step
  Update $\theta, \psi \leftarrow \text{Adam}(\nabla \mathcal{L}_{\text{batch}})$
until convergence

dataset $\mathcal{D}$. For each instance, the forward policy $P_F(\cdot \mid s; \theta)$ generates a trajectory τ by sequentially modifying the state, starting from $s_0 = \mathbf{x}_0$, until termination. At each timestep t, we apply a mask $\mathcal{M}(s_t, \mathbf{x}_0)$ to the policy logits, enforcing domain constraints and preventing re-modification of features within the same trajectory. Actions are sampled in two stages: first selecting a feature index d, then a value v from the valid domain $V(d)$. Upon reaching a terminal state $\mathbf{x}'$, we compute the composite reward $R(\mathbf{x}' \mid \mathbf{x}_0, y')$ (Equation (4)). The policy parameters θ and flow estimator parameters ψ are updated jointly via gradient descent on $\mathcal{L}_{\text{TB}}$ (Equation (6)).

As established in prior work, minimizing this objective drives the implicit sampling distribution of the policy toward the target distribution defined by the reward, $P_F(\mathbf{x}'|\mathbf{x}_0, y') \propto R(\mathbf{x}'|\mathbf{x}_0, y')$. We employ a fixed uniform backward policy P_B, which simplifies the training dynamics by stabilizing the target flow distribution. The reward is computed only at terminal states; all computations are performed in the log domain for numerical stability.

Convergence. To evaluate the stability and convergence of the proposed training procedure, we monitor the Trajectory Balance loss ($\mathcal{L}_{\text{TB}}$) across training iterations. Figure 4 presents the learning curve for the *German Credit* dataset over approximately 400 training steps.

We observe a rapid initial decrease in the loss during the first 50 steps, indicating that the policy quickly learns to satisfy basic domain constraints and identify high-reward regions of the counterfactual space. While the raw loss exhibits some stochasticity, which is typical for GFlowNet training due to the high-dimensional action space and sampling-based updates, the Exponential Moving Average (EMA) shows a consistent downward trend.

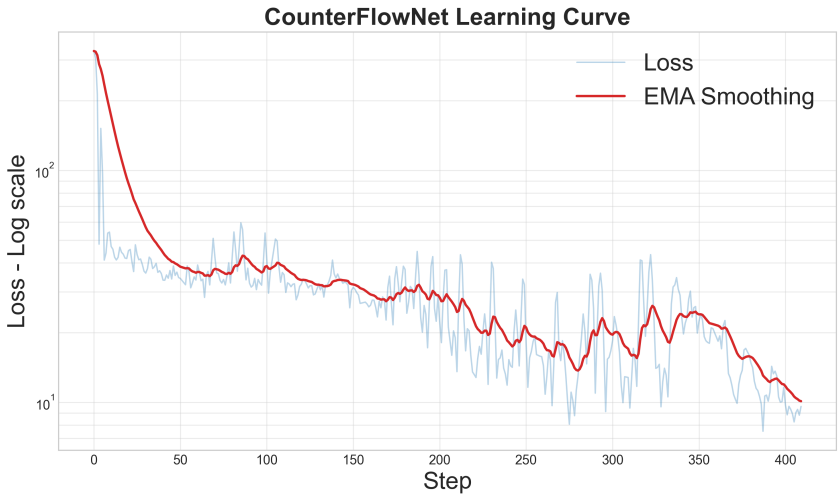


Figure 4: Training loss ($\mathcal{L}_{\text{TB}}$) for the German Credit dataset. The light blue curve represents the raw stochastic updates, while the bold red line shows the Exponential Moving Average (EMA). The logarithmic scale highlights the sustained convergence and numerical stability of the conditioned Trajectory Balance objective.

Table 4: Dataset characteristics. Protocol A: discretized ($B = 4$); Protocol B: continuous with generation-time quantization ($B = 64$).

Dataset	**Protocol**	n	d_{cont}	d_{cat}	**Imbalance**
German Credit	A	1,000	3	17	30%
Adult	A,B	32,561	4	8	24%
Graduate Adm.	A	500	3	4	34%
Student Perf.	A	649	4	10	33%
Lending Club	B	30,000	8	4	22%
GMC	B	30,000	6	3	7%
Bank Marketing	B	30,000	7	9	11%
Default	B	27,000	14	9	22%

A.3 Sampling and Candidate Selection

Generating a pool of candidates and selecting among them according to quality criteria is standard practice across counterfactual generation methods. Mothilal et al. [2020] subject each candidate to per-instance gradient-based optimisation, iteratively refining proposals toward lower loss. Furman et al. [2025] generate a pool via a single forward pass and retain those maximising a composite quality score. In both cases, reported metrics reflect the quality of *selected*, not raw, samples. COUNTERFLOWNET follows the same paradigm. Given $\mathbf{x}_0$ and target class y', we generate K independent trajectories from P_F, discard those for which $h(\mathbf{x}') \neq y'$, and retain the top-N unique candidates ranked by $R(\mathbf{x}' \mid \mathbf{x}_0, y')$. Since P_F samples proportionally to the reward, the generated pool already concentrates on high-quality regions of the counterfactual distribution. Selection further identifies the most representative candidates within it. Crucially, all components of R, including the LOF-based plausibility term and the proximity measure, are computed from training-set statistics only, so candidate selection operates without access to any information derived from the held-out evaluation instances.

B Experimental Setup

B.1 Datasets

Table 4 summarizes the datasets used to train and evaluate COUNTERFLOWNET, including the number of continuous (d_{cont}) and categorical (d_{cat}) features, the target class imbalance, and the assigned evaluation protocol. Below, we describe each dataset individually.

B.1.1 Protocol A Datasets

German Credit. The German Credit dataset [Hofmann, 1994] from the UCI Machine Learning Repository contains information about bank customers applying for credit. Each instance includes 20 attributes covering demographic information (age, personal status, housing), financial indicators (credit amount, savings, checking account status), and employment details. The binary classification task is to assess whether a customer presents a good or bad credit risk. The dataset comprises 1,000 instances with a 30% minority class ratio. Available at: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`.

Adult Income. The Adult Census Income dataset [Kohavi et al., 1996], extracted from the 1994 U.S. Census database, contains demographic information such as age, education, occupation, work hours per week, and capital gain/loss. The binary classification task is to predict whether an individual's annual income exceeds \$50,000. With 32,561 samples and a 24% positive class rate, this dataset is widely used in fairness and explainability research due to the presence of sensitive attributes (race, gender, age). This dataset is used under both Protocol A and Protocol B. Available at: `https://archive.ics.uci.edu/ml/datasets/adult`.

Graduate Admission. The Graduate Admission dataset [Acharya et al., 2019] contains records of Indian students' applications to Master's programs, including GRE scores, TOEFL scores, university rating, statement of purpose strength, letter of recommendation strength, undergraduate GPA, and research experience. The original target is a continuous admission probability; following Vo et al. [2023], we binarize it at a threshold of 0.7 to classify students as having a higher or lower chance of admission. The dataset contains 500 instances. Available at: `https://www.kaggle.com/datasets/mohansacharya/graduate-admissions`.

Student Performance. The Student Performance dataset [Cortez and Silva, 2008] records academic outcomes of secondary school students from two Portuguese schools, with 14 features including demographic attributes (age, family size), social factors (parents' education, family support), and academic indicators (study time, first-period grades). The binary classification task is to predict whether a student achieves a final score above average. The dataset contains 649 instances with train/test splits corresponding to the two schools. Available at: `https://archive.ics.uci.edu/ml/datasets/student+performance`.

B.1.2 Protocol B Datasets

Lending Club. The Lending Club dataset [Jagtiani and Lemieux, 2019] contains detailed information about loans issued through the Lending Club peer-to-peer lending platform. Features include borrower characteristics (credit score, annual income, employment length), loan specifics (loan amount, interest rate, purpose), and performance indicators (payment status, delinquency). The binary classification task is to predict whether a loan will be fully paid or charged off (default). The dataset contains 30,000 samples with a 22% default rate. Available at: `https://www.kaggle.com/datasets/wordsforthewise/lending-club`.

Give Me Some Credit (GMC). The Give Me Some Credit dataset [Pawelczyk et al., 2020] contains anonymized records of credit users with features such as debt-to-income ratio, number of delinquencies, monthly income, age, and number of open credit lines. The target variable indicates whether a user experienced a serious delinquency (more than 90 days overdue) within the previous two years. The dataset contains 30,000 samples with a highly imbalanced 7% positive class rate. Available at: `https://www.kaggle.com/c/GiveMeSomeCredit`.

Bank Marketing. The Bank Marketing dataset [Moro et al., 2014] contains information from a direct marketing campaign conducted by a Portuguese banking institution. Features include client data (age, job, marital status, education), campaign contact information (communication type, day, month),

economic indicators, and previous campaign outcomes. The prediction task is to determine whether a client will subscribe to a term deposit. The dataset comprises 30,000 samples with an 11% positive class rate. Available at: `https://archive.ics.uci.edu/ml/datasets/bank+marketing`.

Credit Default. The Credit Default dataset [Yeh and Lien, 2009] contains information on credit card clients in Taiwan, including demographic factors, credit data, payment history, and bill statements. The target variable indicates whether the client defaulted on their payment in the following month. With 23 features (14 numerical and 9 categorical) and 27,000 samples, this dataset presents complex feature interdependencies common in financial data. Available at: `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`.

B.2 Evaluation Protocols and Discretization

COUNTERFLOWNET operates on discrete action spaces by design; the two protocols differ in where the discretization happens.

Protocol A follows Vo et al. [2023]: continuous features are quantized with the bin ranges released by L2C *before* classifier training, so the classifier, COUNTERFLOWNET, and all baselines operate on the same discrete features.

Protocol B keeps the original classifier on continuous features. Only at the *counterfactual generation* stage do we quantize each continuous feature into $B = 64$ uniform bins (scikit-learn's *KBinsDiscretizer*), and decode each generated bin index back to the continuous domain via its bin center. This yields a per-feature reconstruction error of at most $1/(2B)$ of the feature range, i.e. $1/128$. The quantization enables tractable sequential generation, provides a tunable proximity-diversity trade-off via B (Section C.2), and matches the implicit discretization in competitor methods (e.g., DiCoFlex's L_p neighbor selection operates on finite training examples).

B.3 Evaluation Metrics

This section provides detailed descriptions of the evaluation metrics used in our experiments, organized by protocol. The two protocols are adopted from two different works, Vo et al. [2023] (Protocol A) and Furman et al. [2025] (Protocol B). The main difference between them is that Protocol A assumes all features are discretized, while Protocol B works with both continuous and categorical features. Furthermore, Protocol A reports metrics normalized to percentages. Table 5 outlines the metrics for Protocol A, while Table 6 details the metrics for Protocol B.

B.3.1 Modes

Modes adapts the mode-counting metric of the GFLOWNET literature [Bengio et al., 2021], which counts greedily discovered candidates whose reward exceeds a threshold and that are dissimilar to all previously discovered modes. Because the counterfactual reward $R(\mathbf{x}' \mid \mathbf{x}_0, y')$ is conditional, each original instance defines its own reward landscape, so we count modes within each group g and average over groups, as diversity is evaluated per input in the counterfactual literature [Mothilal et al., 2020].

For group g with counterfactual set $\mathcal{C}_g$, let $\mathcal{C}_g^+ = \{\mathbf{x}' \in \mathcal{C}_g : h(\mathbf{x}') = y',\ R(\mathbf{x}' \mid \mathbf{x}_0, y') > \tau_R\}$. Visiting $\mathcal{C}_g^+$ in order of decreasing reward, starting from $\mathcal{M}_g = \emptyset$, a counterfactual $\mathbf{x}'$ is added to $\mathcal{M}_g$ if and only if $d(\mathbf{x}', \mathbf{m}) > \tau_d$ for every $\mathbf{m} \in \mathcal{M}_g$, where

$$d(\mathbf{a}, \mathbf{b}) = \frac{1}{D}\Big(\sum_{j \in \mathcal{N}} |\beta_j(a_j) - \beta_j(b_j)| + \sum_{j \in \mathcal{C}} \mathbb{1}[a_j \neq b_j] \Big),$$

and β_j maps a numerical value to its bin index under the $B = 64$ discretization. Groups with no qualifying CF contribute $|\mathcal{M}_g| = 0$. We use $\tau_R = 0.5$, the midpoint of the $[0, 1]$ reward, and $\tau_d = 0.7$.

R is the Protocol B reward with the dataset-specific weights of Table 7, applied identically to all methods. Because $|\mathcal{M}_g|$ grows with the number of CFs per group, all methods are compared at an equal budget of 25 CFs per instance.

Table 5: Description of quantitative evaluation metrics for Protocol A. For each test instance $i \in \{1, \ldots, N\}$, we generate a set of K counterfactuals $\mathcal{C}_i = \{\mathbf{x}_i'^{(k)}\}_{k=1}^{K}$. Let h denote the classifier, y_i' the desired target class, D the total number of features, and $\mathcal{F}_m$ the set of features subject to monotonic constraints.

Desideratum	**Metric**	**Description**	**Formula**
Validity (↑)	Val.(%)	Proportion of valid counterfactuals.	$\frac{1}{N \cdot K} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}\left[h(\mathbf{x}_i'^{(k)}) = y_i'\right]$
	Cov.(%)	Proportion of instances with at least one valid CF.	$\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[\exists\, k : h(\mathbf{x}_i'^{(k)}) = y_i'\right]$
Sparsity (↑)	Spars.(%)	Proportion of features that remain unchanged.	$\frac{1}{N \cdot K} \sum_{i=1}^{N} \sum_{k=1}^{K} \frac{1}{D} \sum_{j=1}^{D} \mathbb{1}\left[x_{i,j}'^{(k)} = x_{i,j}\right]$
Diversity (↑)	Div.(%)	Mean pairwise Hamming distance.	$\frac{1}{N} \sum_{i=1}^{N} \frac{2}{K(K-1)} \sum_{k<l} \frac{1}{D} \sum_{j=1}^{D} \mathbb{1}\left[x_{i,j}'^{(k)} \neq x_{i,j}'^{(l)}\right]$
Balance (↑)	H. Mean(%)	Harmonic mean of Diversity and Sparsity.	$\frac{2 \cdot \text{Div} \cdot \text{Spars.}}{\text{Div} + \text{Spars.}}$
Constraints (↑)	Unary(%)	Proportion of CFs satisfying monotonic constraints.	$\frac{1}{N \cdot K} \sum_{i=1}^{N} \sum_{k=1}^{K} \frac{1}{\lvert\mathcal{F}_m\rvert} \sum_{j \in \mathcal{F}_m} \mathbb{1}\left[\text{mon}_j(\mathbf{x}_i'^{(k)}, \mathbf{x}_i)\right]$

B.4 Baselines

We compare COUNTERFLOWNET against a comprehensive set of state-of-the-art methods, spanning three major paradigms in counterfactual explanation generation.

Optimization-based methods generate CFs by optimizing an objective function for each input instance individually.

- **DiCE** [Mothilal et al., 2020] is a prominent method that generates a diverse set of CFs by adding a Determinantal Point Process (DPP) term to the optimization objective. It handles both continuous and categorical features. For Protocol A, we use the random search (**DICE-R**) and genetic algorithm (**DICE-G**) variants. For Protocol B, we use the KD-Tree-based search.
- **COPA** [Bui et al., 2022] generates CFs for structured policies and has been adapted for tabular data. It is included in Protocol A.

Sampling-based methods explore the feature space to find CFs.

- **MCCE** [Redelmeier et al., 2024] builds a local model around the instance to be explained and then uses Monte Carlo sampling from a conditional distribution of mutable features to generate CFs that lie on the data manifold. It is evaluated under Protocol A.

Generative and Amortized methods train a single model to generate CFs for any input instance in a single forward pass, amortizing the computational cost.

- **L2C** [Vo et al., 2023] is an amortized framework that learns feature-based perturbation and selection distributions to generate diverse and privacy-preserving CFs on discretized data. It is a key baseline in Protocol A.

Table 6: Description of quantitative evaluation metrics for Protocol B. We generate N counterfactuals $\{\mathbf{x}'_n\}_{n=1}^N$ for a test set of N instances $\{\mathbf{x}^0_n\}_{n=1}^N$, of which N_{val} are valid. Let h denote the classifier, D_{cat} and D_{num} the number of categorical and continuous features, $D = D_{\text{cat}} + D_{\text{num}}$ the total, V_j the range of feature j, G the number of unique original instances (groups), K_g the number of counterfactuals for group g, and $\mathcal{N}$, $\mathcal{C}$ the sets of numerical and categorical feature indices.

Desideratum	**Metric**	**Description**	**Formula**
Validity (↑)	Val.	Success rate of changing model predictions.	$\frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left[h(\mathbf{x}'_n) = y'\right]$
Proximity (↓)	Prox.-Cont	L_1 distance on continuous features.	$\frac{1}{N_{\text{val}}}\sum_{n=1}^{N_{\text{val}}} \left\|\mathbf{x}^0_{n,\text{num}} - \mathbf{x}'_{n,\text{num}}\right\|_1$
Sparsity (↓)	Spars.-Cat	Proportion of changed categorical features.	$\frac{1}{N_{\text{val}}}\sum_{n=1}^{N_{\text{val}}} \frac{\left\|\mathbf{x}^0_{n,\text{cat}} - \mathbf{x}'_{n,\text{cat}}\right\|_0}{D_{\text{cat}}}$
	ϵ-Spars.	Continuous features changed beyond $\epsilon \cdot V_j$ (ϵ=0.05).	$\frac{1}{N_{\text{val}}}\sum_{n=1}^{N_{\text{val}}} \frac{1}{D_{\text{num}}}\sum_{j=1}^{D_{\text{num}}} \mathbb{1}\left[\lvert x^0_{n,j} - x'_{n,j}\rvert > \epsilon V_j\right]$
Plausibility (↓)	LOF	Median Local Outlier Factor (log scale).	$\text{median}_{n=1}^{N_{\text{val}}} \ \log\left(\text{LOF}_k(\mathbf{x}'_n)\right)$
Diversity (↑)	Div.	Mean pairwise mixed distance between counterfactuals, averaged over groups.	$\frac{1}{G}\sum_{g=1}^{G} \frac{1}{D\binom{K_g}{2}}\sum_{i<j}^{K_g}\Big(\left\|\mathbf{x}'_{i,\text{num}} - \mathbf{x}'_{j,\text{num}}\right\|_1 + \left\|\mathbf{x}'_{i,\text{cat}} - \mathbf{x}'_{j,\text{cat}}\right\|_0\Big)$
	Modes	Mean number of valid, high-reward, mutually distant CFs per group (Appendix B.3.1).	$\frac{1}{G}\sum_{g=1}^{G} \lvert\mathcal{M}_g\rvert$

- **C-CHVAE** [Pawelczyk et al., 2020] uses a Conditional Variational Autoencoder (CVAE) to learn class-conditional latent representations, generating plausible CFs by searching in the latent space. It is evaluated under Protocol B.
- **DiCoFlex** [Furman et al., 2025] employs a conditional normalizing flow to generate diverse CFs in a single forward pass. It allows for inference-time control over sparsity and actionability constraints. It is a key baseline in Protocol B.
- **LAPACE** [Jiang et al., 2026] trains a label-conditional Gaussian-mixture VAE on the predicted labels of the classifier and generates CFs by interpolating an instance's latent code toward the centroids of the target-class mixture components. As it has no implementation for this benchmark, we re-implement it on our classifier, without constraint correction. It yields one CF per target-class component (25 components per class, matching the 25 CFs per instance of Table 2): the first point along the interpolation path at which the classifier's prediction flips, or the centroid if none does.

B.5 Implementation Details

Our implementation builds upon the code released with Koziarski et al. [2024], which is publicly available at https://github.com/koziarskilab/RGFN.

Architecture. Both policy networks, modeling actions and flow respectively, are implemented as multilayer perceptrons (MLPs) with two hidden layers of width 256 and ReLU activation function.

Training. We optimize COUNTERFLOWNET using the Trajectory Balance objective (Eq. 11) with the Adam optimizer and a learning rate of 0.005. For each dataset, training uses a batch of 1000 factual points sampled sequentially. Moreover, the reward temperature is consistently fixed at $\beta = 40.0$, and we set the *logZ_multiplier* to 10.

Reward Components. Table 7 lists the reward weights, validity margin ϵ, and bin count B used for each dataset. Protocol A uses a simplified reward with only validity and sparsity terms. For all datasets, the proximity component is computed over numerical features and the sparsity component over categorical ones. The Lending Club exception and its rationale are described in Section 4.

Table 7: Reward configuration per protocol and dataset.

Protocol	Datasets	λ_v	λ_s	λ_p	λ_d	ϵ	B
A	German Credit, Adult, Graduate Adm., Student Perf.	1	0.01	0	0	0.1	4
B	Adult, Bank Marketing, Default, GMC	1	0.8	0.4	0.4	0.1	64
B	Lending Club	1	0.5	0.4	0.3	0.25	64

Generation. For each test instance, we sample $K = 25$ CFs via forward policy rollouts. All experiments in Protocol A are repeated over 5 random seeds, while those in Protocol B use a single seed.

Hardware. All training and evaluation of COUNTERFLOWNET were performed on a single NVIDIA GeForce RTX 4090 GPU. Each dataset required less than 10 hours of training (Section C.5).

C Additional Results

C.1 Generator Ablation: Search and Reinforcement Learning

This ablation tests whether the diversity of COUNTERFLOWNET comes from the GFLOWNET objective or from the MDP and reward it shares with simpler methods. We keep the action space, reward, classifier, and $B = 64$ discretization of COUNTERFLOWNET fixed and replace only the generator. *Random*, *beam*, and *evolutionary* search are training-free searches over the same action space. *REINFORCE* uses the policy network, sampler, and candidate selection of COUNTERFLOWNET, but is trained to maximize expected reward with REINFORCE (1000 trajectories per iteration, as for COUNTERFLOWNET) instead of the Trajectory Balance objective (Eq. 11).

All methods use the reward configuration of Table 7 and the instances of the COUNTERFLOWNET runs behind Table 2. Metrics are computed as in Table 2.

COUNTERFLOWNET offers the most modes per instance on every dataset (Table 8). Beam search, the strongest baseline, matches its validity on most datasets and has better ϵ-sparsity, but concentrates on a few solutions per instance: sharing the action space and reward is not sufficient for diversity. Random and evolutionary search reach the highest diversity through large, undirected edits with poor proximity and plausibility, and most of their CFs fall below the reward threshold of the Modes metric. REINFORCE collapses early in training to a near-deterministic policy that edits the same feature set on every instance, and therefore offers few modes and little diversity. As it uses the hyperparameters of COUNTERFLOWNET, a batch-mean baseline, and no entropy regularization, it is a lower bound for reinforcement learning.

C.2 Discretization Granularity

We vary the number of bins $B \in \{4, 8, 16, 32, 64, 128\}$ on Adult under Protocol B, keeping all other hyperparameters at the Adult configuration of Table 7. $B = 64$ corresponds to the main run of Table 2. Figure 5 shows the results.

Table 8: COUNTERFLOWNET (CFN) against search and REINFORCE baselines that share its action space, reward, and classifier. Best per dataset in **bold**.

Method	Val. ↑	Prox.-Cont ↓	Spars.-Cat ↓	ϵ-Spars. ↓	LOF ↓	Div. ↑	Modes ↑
			Adult Income				
CFN	**1.00**	0.19	0.27	0.37	**0.19**	0.21	**3.69**
REINFORCE	0.05	**0.00**	0.50	**0.00**	0.39	0.00	0.52
Random search	0.60	1.87	0.27	0.37	0.47	**0.79**	1.18
Beam search	**1.00**	0.29	**0.24**	0.18	**0.19**	0.16	1.06
Evol. search	0.65	2.15	0.58	0.60	0.50	0.67	0.35
			Bank				
CFN	**1.00**	0.25	0.23	0.38	0.21	0.21	**8.98**
REINFORCE	0.14	**0.20**	**0.11**	**0.14**	0.30	0.00	0.80
Random search	0.82	2.23	0.26	0.32	0.40	**1.07**	1.22
Beam search	**1.00**	0.25	0.24	**0.14**	**0.10**	0.10	1.10
Evol. search	0.94	2.30	0.53	0.49	0.44	0.72	0.51
			Default				
CFN	**0.96**	0.12	0.26	0.19	0.34	0.14	**4.94**
REINFORCE	0.23	**0.00**	0.56	**0.00**	0.48	0.00	0.85
Random search	0.16	1.17	**0.24**	0.17	0.81	**0.76**	0.28
Beam search	0.94	0.02	0.37	0.04	**0.24**	0.07	0.97
Evol. search	0.18	1.72	0.39	0.27	0.83	0.52	0.13
			GMC				
CFN	**1.00**	0.12	0.64	0.25	**0.23**	0.18	**4.26**
REINFORCE	0.71	**0.00**	1.00	**0.00**	0.31	0.00	0.97
Random search	0.99	5.22	**0.34**	0.37	1.41	**3.44**	0.18
Beam search	**1.00**	0.04	0.76	0.08	0.30	0.17	1.84
Evol. search	0.86	5.53	0.59	0.49	1.20	3.08	0.22
			Lending Club				
CFN	**0.91**	0.55	0.41	0.30	0.21	0.24	**9.23**
REINFORCE	0.19	0.44	0.75	**0.12**	**0.16**	–	0.59
Random search	0.07	0.78	**0.34**	0.28	0.32	**0.65**	0.59
Beam search	0.79	**0.42**	0.54	0.17	**0.16**	0.14	2.56
Evol. search	0.25	1.23	0.65	0.49	0.39	0.40	0.41

As B increases, the finer-grained action space yields more plausible CFs closer to the data manifold (decreasing LOF), but at the cost of worse proximity and ϵ-sparsity due to accumulated small adjustments. Diversity peaks at lower B, suggesting that coarser grids force exploration of structurally different solutions. Validity remains high in all settings, reaching its peak at $B = 64$. Our choice of $B = 64$ represents a balanced compromise between plausibility and other desiderata.

C.3 Reward Components

We perform an extensive ablation study to analyze the effect of the reward component weights on the generated CFs. The composite reward (Eq. 4) is controlled by four weights: λ_v (validity), λ_d (proximity), λ_p (plausibility), and λ_s (sparsity). In our experiments, we keep $\lambda_v = 1.0$ and $\lambda_s = 0.4$ fixed and perform a grid search over λ_d and λ_p on the Adult dataset.

The results are visualized in Figure 6. The quantitative results confirm the trade-offs discussed in the main paper. Specifically, we observe that decreasing the proximity weight (λ_d) consistently leads to an increase in diversity, but also a worsening of proximity (L1), sparsity (ϵ-sparsity), and plausibility (LOF). Similarly, decreasing the plausibility weight (λ_p) improves proximity and sparsity at the expense of plausibility. For instance, with $\lambda_d = 1.0$, decreasing λ_p from 1.0 to 0.0 reduces the L1 distance, but increases the LOF score.

C.4 Joint Sensitivity to Discretization and Reward Coefficients

Appendices C.2 and C.3 vary the bin count B and the reward weights $\boldsymbol{\lambda}$ one at a time. Here we test whether the two interact, i.e. whether the effect of the reward weights depends on B. We retrain COUNTERFLOWNET from scratch on Adult for every cell of the grid $B \in \{16, 64\} \times \lambda_d \in \{0, 0.4, 1.0\} \times \lambda_p \in \{0, 0.4, 1.0\}$, with $\lambda_v = 1.0$ and $\lambda_s = 0.8$ fixed at the Protocol B values (Table 7). All other hyperparameters follow the Protocol B Adult configuration. Each cell is a single

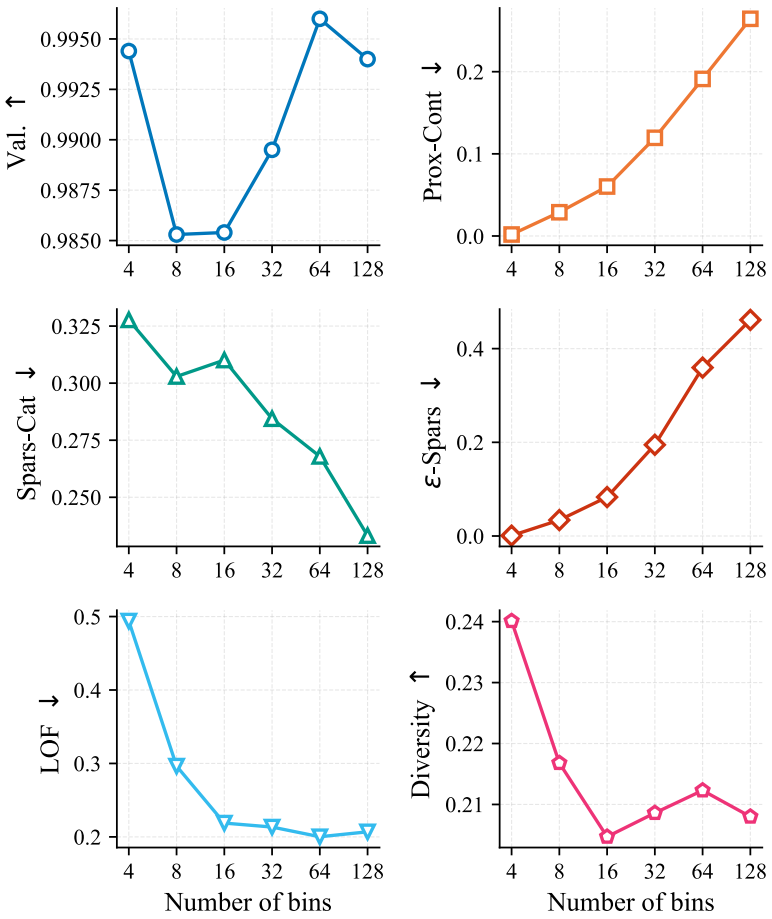
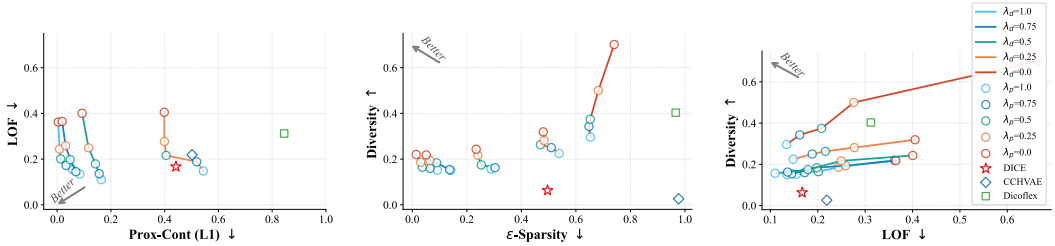


Figure 5: Effect of discretization granularity B on Adult dataset.



(a) Proximity vs Plausibility (b) Diversity vs Sparsity (c) Diversity vs Plausibility

Figure 6: Ablation of λ_p and λ_d with fixed weights for other reward weights ($\lambda_v = 1.0, \lambda_s = 0.4$)

training run evaluated on the validation split on all sampled CFs, without the reward-based candidate selection.

B and $\boldsymbol{\lambda}$ act on different parts of the model. The bin count sets the resolution of the action space and of the decoded feature values, a property of the forward policy P_F, whereas the reward weights set the shape of the target distribution, a property of R. We therefore expect the reward trade-offs to look the same for both bin counts, and Figure 7 confirms this: in most panels the two lines respond to the reward weights in the same direction. To quantify this, we rank the nine reward settings by each metric separately for each bin count and compute the Spearman rank correlation between the two rankings. The correlation is perfect for validity, high for proximity and plausibility (0.88), and somewhat lower for ϵ-sparsity (0.80) and diversity (0.75). The reward setting that is best for validity,

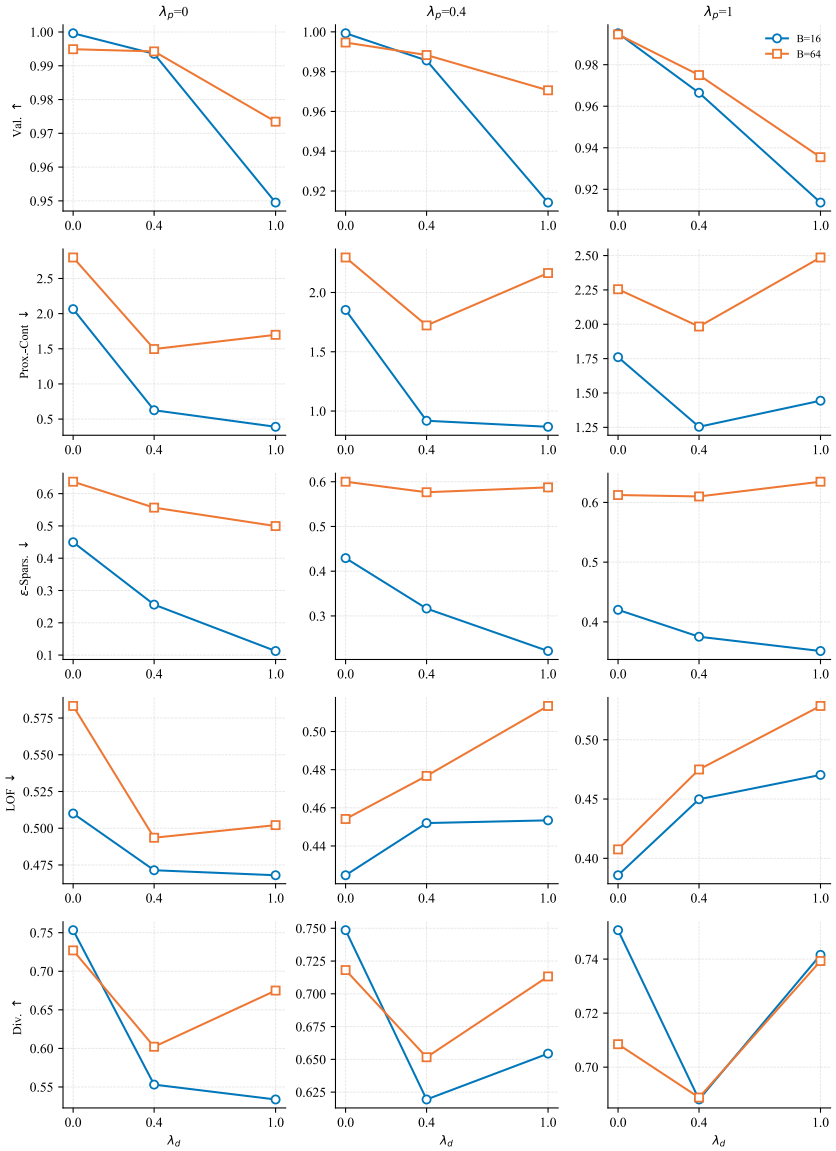


Figure 7: Joint sensitivity to B and the reward weights on Adult (λ_v=1.0, λ_s=0.8, validation split). Rows: metrics; columns: plausibility weight λ_p; x-axis: proximity weight λ_d; one line per bin count B.

ϵ-sparsity, and plausibility is the same for both bin counts. For proximity, the best setting moves by one grid step towards a smaller proximity weight at the finer grid. For diversity, the best setting changes, but at the finer grid the competing settings are within a few percent of each other. Validity remains above 91% everywhere. The bin count mainly shifts the overall level of the metrics: the finer grid yields larger proximity distances and worse ϵ-sparsity in every setting, consistent with Appendix C.2, while the effect of the reward weights is preserved. The reward weights can therefore be chosen independently of the bin count.

C.5 Training and Inference Runtime

Table 9 reports the training time and the inference time per generated CF of every method on the Protocol B datasets. The methods were timed on different machines with their own timers, so the table compares orders of magnitude rather than exact ratios. Training COUNTERFLOWNET is a one-time cost of under 10 hours per dataset, comparable to DiCoFlex and higher than for the

VAE-based C-CHVAE and LAPACE, while DiCE and the searches need no training. At inference, COUNTERFLOWNET generates a CF in milliseconds, on the same order as DiCE, C-CHVAE, DiCoFlex, the searches, and REINFORCE, which shares its policy network; LAPACE, which decodes in a single forward pass, is about an order of magnitude faster.

Table 9: Training time (s) and inference time (ms per generated CF) on the Protocol B datasets.

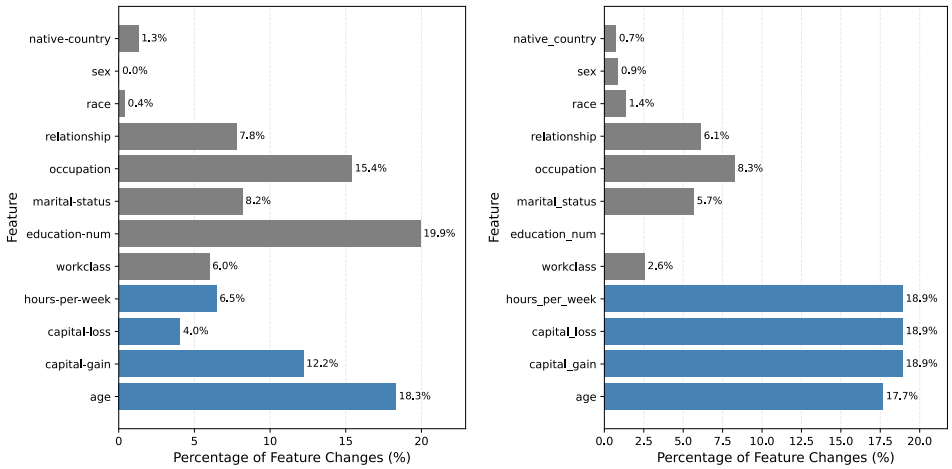
Method	**Adult**	**Bank**	**Default**	**GMC**	**Lending Club**
Training (s)					
CFN	26692	28651	35576	24423	20740
DICE	n/a	n/a	n/a	n/a	n/a
CCHVAE	36	26	36	37	34
DiCoFlex	31856	32012	38972	26942	25341
LAPACE	4872	3191	1956	2452	3844
REINFORCE	1670	8285	18120	10864	9586
Random / beam / evol. search	n/a	n/a	n/a	n/a	n/a
Inference (ms per CF)					
CFN	13.9	15.0	18.9	12.6	11.6
DICE	13.9	10.2	16.9	5.9	7.9
CCHVAE	30.1	57.6	60.8	22.3	17.6
DiCoFlex	3.0	6.3	8.2	8.1	6.0
LAPACE	0.9	1.1	2.0	1.3	0.8
REINFORCE	15.7	17.1	83.5	102.4	10.3
Random search	5.4	7.4	5.8	4.0	3.0
Beam search	17.1	17.3	23.7	13.9	12.9
Evol. search	6.7	2.8	4.5	2.8	2.1

C.6 Feature Utilization

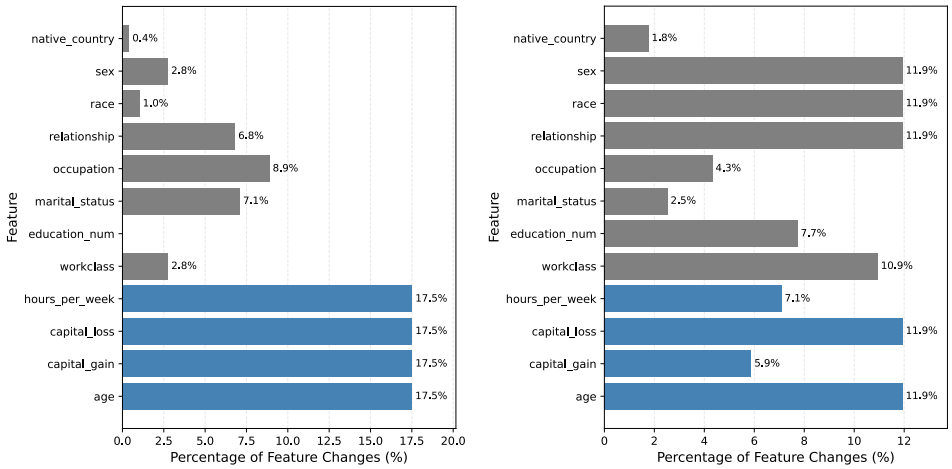
We analyze the feature utilization of COUNTERFLOWNET and baseline methods on the Adult dataset to understand which features are most frequently modified to generate CFs. Figure 8 illustrates the percentage of changes for each feature. COUNTERFLOWNET (Figure 8a) demonstrates a balanced utilization of both numerical and categorical features. The most frequently modified features include ‘education-num‘ (19.9%), ‘age‘ (18.3%), and ‘occupation‘ (15.4%), indicating that COUNTERFLOWNET leverages a diverse set of attributes to find recourse. In contrast, DiCE (Figure 8b) and CCHVAE (Figure 8c) exhibit a strong bias towards numerical features, with changes heavily concentrated on ‘age‘, ‘capital-gain‘, ‘capital-loss‘, and ‘hours-per-week‘ (all above 17%). This suggests that these methods may overlook viable CFs that involve changes to categorical attributes. DiCoFlex (Figure 8d) shows a more uniform, yet still skewed, distribution, with a wider range of features being modified but still a noticeable preference for certain features over others.

C.7 Qualitative Examples

This section provides qualitative examples of CFs generated by COUNTERFLOWNET for the Adult (Table 10) and GMC (Table 11) datasets. For each dataset, we present one instance, showing the original input and two diverse, valid CFs that change the prediction from low income to high income (Adult) or from high risk to low risk (GMC). Changed features are highlighted in **bold**.



(a) COUNTERFLOWNET (b) DiCE



(c) CCHVAE (d) DiCoFlex

Figure 8: Visualization of feature utilization of COUNTERFLOWNET method and other baseline methods on the Adult dataset. The plots show the frequency with which each feature is modified in the generated CFs. COUNTERFLOWNET demonstrates a more balanced feature utilization compared to baselines.

Table 10: Counterfactual examples for the Adult dataset. The original prediction is "low income," and the counterfactual prediction is "high income."

Feature	**Original Input**	**Counterfactual 1**	**Counterfactual 2**
Age	25	25	**29**
Workclass	Self-emp-not-inc	**Federal-gov**	Self-emp-not-inc
Education	HS-grad	**Prof-school**	HS-grad
Marital Status	Never-married	**Married-AF-spouse**	**Married-civ-spouse**
Occupation	Sales	Sales	Sales
Relationship	Other-relative	**Wife**	**Wife**
Race	Asian-Pac-Islander	Asian-Pac-Islander	Asian-Pac-Islander
Sex	Male	Male	Male
Capital Gain	781	781	**7031**
Capital Loss	34	34	34
Hours per Week	50	50	50
Native Country	United-States	United-States	United-States

Table 11: Counterfactual examples for the GMC dataset. The original prediction is "high risk," and the counterfactual prediction is "low risk."

Feature	Original Input	Counterfactual 1	Counterfactual 2
Revolving Util. of Unsecured Lines	69	69	69
Age	45	**21**	**56**
Times 30-59 Days Past Due	1	**0**	**3**
Debt Ratio	162.57	162.57	162.57
Monthly Income	5488.28	5488.28	5488.28
Open Credit Lines & Loans	5	**0**	**4**
Times 90+ Days Late	0	0	0
Real Estate Loans/Lines	0	0	0
Times 60-89 Days Past Due	0	0	**3**
Number of Dependents	0	0	0

D Limitations

COUNTERFLOWNET is designed for tabular data with mixed feature types, the most common setting for high-stakes counterfactual explanations; extension to text, images, or time series would require modality-specific encoders and remains future work. Our unified discrete action space requires quantizing continuous features, introducing a bounded reconstruction error (Section B.2). Training entails a one-time cost of 5–10 hours per dataset, offset by fast per-instance inference suitable for interactive use (Section C.5). Finally, like all counterfactual methods, COUNTERFLOWNET can in principle expose decision-boundary information to adversaries [Aïvodji et al., 2020], and inherits any biases present in the underlying classifier; integrating fairness-aware reward components is a natural extension we leave to 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 abstract and Section 1 state our four contributions (reward-proportional CF sampling via GFlowNets, mixed-feature support via two-stage actions, inference-time actionability masking, and broad empirical evaluation), each of which is substantiated in Sections 3 and 4 and the corresponding ablations.

   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: A dedicated Limitations section (Appendix D) discusses the restriction to tabular data, the bounded reconstruction error introduced by quantizing continuous features, the per-dataset training cost, and the inherited risks of decision-boundary leakage and classifier bias.

   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: [N/A]

   Justification: The paper does not introduce new theorems. We rely on the established reward-proportional sampling property of GFlowNets [Bengio et al., 2021, Malkin et al., 2022] and reproduce the flow-matching and Trajectory Balance derivations in Appendix A.1 for self-containment.

   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: Sections 3 and 4 fully describe the MDP, two-stage policy, composite reward, and training objective; Appendix A.2 gives the full training algorithm; Appendix B.5 reports architecture, optimizer and learning rate, reward weights $(\lambda_v, \lambda_d, \lambda_p, \lambda_s)$, the discretization bin count, and the generation procedure, and Appendix B.2 the per-protocol preprocessing.

   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: All eight benchmark datasets are publicly available (UCI, Kaggle), with URLs provided in Appendix B.1. Our code, including training and evaluation scripts, is publicly available at `https://github.com/ofurman/CounterFlowNet`; our implementation extends the public RGFN codebase [Koziarski et al., 2024] (Appendix B.5).

   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: Appendix B.5 specifies the optimizer (Adam, lr 0.005), batch size, reward temperature, $\log Z$ multiplier, MLP architecture (two hidden layers of 256 with ReLU), reward weights per dataset, the number of sampled CFs ($K{=}10$), and seed counts (5 for Protocol A, 1 for Protocol B); Appendix B.2 gives the per-protocol preprocessing and discretization. Evaluation protocols and metrics are detailed in Appendix B.3.

   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: Protocol A results in Table 1 are averaged over 5 random seeds (the source of variability is policy initialization and trajectory sampling); we report mean values for table compactness, and the per-dataset gaps between COUNTERFLOWNET and the next-best baseline (e.g., ≥ 5 percentage points on Harmonic Mean for German Credit, Adult, and Student Performance) are well above typical run-to-run variance we observed during development. Protocol B follows the single-seed evaluation convention of the source benchmark [Furman et al., 2025] for direct comparability with reported baseline numbers.

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: Appendix B.5 states that training and evaluation were performed on a single NVIDIA GeForce RTX 4090 GPU, and Appendix C.5 (Table 9) reports per-dataset training time (21k–36k seconds, $<$10h) and per-CF inference time (12–19 ms) of COUNTERFLOWNET on the Protocol B datasets, alongside the measured runtimes and hardware of the baselines.

   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 uses only publicly available, de-identified tabular benchmarks, involves no human subjects or crowdsourcing, and aims at improving transparency and recourse for algorithmic decisions, in line with 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: [Yes]

Justification: Positive impacts (transparency, GDPR-aligned algorithmic recourse in finance, healthcare, and legal decision-making) are discussed in Section 1; potential negative impacts (decision-boundary leakage to adversaries [Aïvodji et al., 2020] and inheritance of classifier bias) and possible mitigations (fairness-aware reward components) are discussed in Appendix D.

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: The paper releases neither generative models with high misuse risk (e.g., pretrained language or image generators) nor any scraped data; the released artifact is a counterfactual-explanation training pipeline for tabular classifiers on already-public benchmarks.

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 datasets are cited with their original sources and URLs in Appendix B.1 (UCI Machine Learning Repository and Kaggle), and the RGFN codebase we build upon [Koziarski et al., 2024] is publicly released and cited in Appendix B.5; baseline methods (DiCE, COPA, MCCE, L2C, C-CHVAE, DiCoFlex, LAPACE) are credited at their respective citations.

    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: [Yes]

    Justification: The code released at `https://github.com/ofurman/CounterFlowNet` includes a README with environment, installation, training, and evaluation instructions, configuration files for each dataset and protocol, and pointers to the public dataset sources documented in Appendix B.1.

    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: The paper does not involve any crowdsourcing or research with human subjects; all experiments are conducted on publicly available, anonymized tabular datasets.

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: The paper does not involve human subjects or any data collection from participants, so IRB review is not applicable.

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: LLMs were used solely for minor language polishing and formatting and are not part of the core methodology, training pipeline, or evaluation of COUNTERFLOWNET.

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.