COUNTERFLOW: TARGET-CONDITIONED FLOW MATCHING FOR VISUAL COUNTERFACTUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

In eXplainable AI, counterfactual explanations have gained significant traction by offering users direct, intuitive, and actionable insights into model decisions. We introduce CounterFlow, a novel framework based on Conditional Flow Matching that generates visual counterfactual explanations by directly steering a source image toward a user-specified target class. By leveraging target-conditioned vector fields, CounterFlow identifies highly realistic semantic modifications while requiring substantially lower computational overhead than traditional GAN- or diffusion-based architectures. Empirical evaluations on standard benchmarks demonstrate that CounterFlow achieves state-of-the-art performance in both visual realism and target class switching. By working directly in the raw pixel space, CounterFlow offers a more practical solution for real-world deployment.

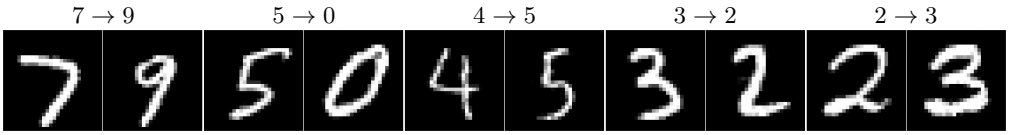
1 INTRODUCTION

Context. On the one hand, in generative modeling, diffusion models currently dominate the landscape for high-quality image, video, and audio synthesis (Rombach et al., 2022; Ho et al., 2022; Liu et al., 2023a). However, they are also well-known for their slow training times and their stochastic sampling paths. To address these limitations, Lipman et al. (2023); Albergo & Vanden-Eijnden (2023); Liu et al. (2023b) introduced Conditional Flow Matching (CFM). CFM is a generative modeling framework based on Continuous Normalizing Flows (CNFs) that uses standard regression objectives for training. Moreover, instead of relying on diffusion processes, CFM generates examples by integrating the CNF's ODE along deterministic trajectories. On the other hand, the generative models' ability to generate complex data distributions and thus to produce realistic examples is particularly appealing for eXplainable Artificial Intelligence (XAI, Dwivedi et al., 2023), where one aims to produce counterfactual explanations (Wachter et al., 2017), *i.e.*, examples *(i)* close to a given input, *(ii)* plausible, and *(iii)* that change the model's prediction. Unlike adversarial examples that rely on imperceptible noise to fool the classifier, counterfactual examples expose meaningful changes, helping users to trust the model while enabling developers to debug it. Our work leverages CFM to create counterfactual explanations: rather than generating images from random noise, our approach transports an input sample toward a counterfactual conditioned on a user-defined target class, while jointly optimizing for plausibility and proximity; see Figure 1.

Related Work. Prior work on image counterfactual explanations falls into two categories: instance-based and generative approaches. Instance-based methods (see *e.g.*, Goyal et al., 2019; Keane & Smyth, 2020) select examples from the training set using similarity metrics to modify the input sample $\mathbf{x}$. However, they require access to the training set at inference and cannot tailor a minimal, plausible edit to a specific example $\mathbf{x}$. We focus instead on generative approaches, a more recent line of work (Khorram & Fuxin, 2022; Jeanneret et al., 2024; Weng et al., 2024; Augustin et al., 2022; Cao et al., 2025) that produces more plausible examples, which we group by generative backbone: GANs, diffusion models, or CFM.

- **GAN.** C3LT (Khorram & Fuxin, 2022) encodes the input (with a VAE encoder) into a latent space, in which it learns two cycle-consistent transformations per pair of classes (y, y') before decoding with the generator of a GAN. It thus requires one model set per pair, impractical for many classes, and its fidelity is bounded by the VAE's reconstruction quality.
- **Diffusion.** DiME (Jeanneret et al., 2024) generates a counterfactual by corrupting the input and denoising it through the backward diffusion process (an inner loop of several steps); at each outer

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



(a) MNIST



(b) CIFAR10

Figure 1: Qualitative counterfactual explanations generated by **CounterFlow** on (a) MNIST and (b) CIFAR10. We show five examples per dataset as *source*/*counterfactual* pairs (*source* on the left of each pair, the corresponding *counterfactual* generated by CounterFlow on the right), and we indicate the source→target class transition above each pair. CounterFlow preserves source-specific style, such as stroke thickness and digit slant on MNIST, or object color and orientation on CIFAR10, modifying only what is needed to reach the target class.

step, the classifier gradient computed on this denoised estimate refines the corrupted instance, and the process repeats until it converges to a valid counterfactual. FastDiME (Weng et al., 2024) instead computes this gradient directly from the model's one-step denoised estimate, bypassing the inner loop entirely. ACE (Jeanneret et al., 2023) takes a different route: it builds an adversarial attack against the classifier and lets the diffusion model's own denoising process project this attack back onto the manifold of realistic images, turning an imperceptible perturbation into a semantically meaningful one. DVCE (Augustin et al., 2022) starts the reverse diffusion from a partially noised version of the input, rather than pure noise, and steers the process by projecting the gradient of an adversarially robust classifier onto a cone around the target classifier's own gradient direction whenever the two disagree, avoiding non-semantic changes. More recently, MaskDiME (Guo et al., 2026) improves on DiME's efficiency and spatial precision with a training-free, adaptive dual-mask mechanism guided by classifier gradients that restricts updates to a localized region rather than the whole image. Regardless of the specific guidance scheme, diffusion-guidance methods must reapply their guidance at every one of many denoising steps, making inference slow: DiME's inner diffusion loop scales quadratically with the number of diffusion steps, and even FastDiME's one-step approximation, despite yielding a roughly 20× empirical speedup, only reduces this to a linear scaling and remains markedly slower than a single forward pass. ACE and DVCE compound this same per-step cost further, respectively adding an adversarial attack and a robust-classifier gradient computation at every step.

- **CFM.** LeapFactual (Cao et al., 2025) builds on Conditional Flow Matching (CFM) to generate counterfactuals. It learns a class-conditional generative flow from a Gaussian prior to the latent space of a pretrained VAE, and repurposes this flow for counterfactual generation through a *lift-and-land* procedure. Given an input, its VAE encoding is first *lifted* toward the prior under the input class by solving the learned ODE backward, and then *landed* back into the latent space under the target class by solving the ODE forward. This "leap" must be repeated a dataset-dependent number of times, with each leap requiring a VAE decoding and a classifier query.
  Our work also builds on CFM, but uses it in a fundamentally different way. We directly learn a transport from an input image to its counterfactual. This avoids (i) the need for multiple costly leaps, (ii) the intermediate traversal through a prior space, and (iii) VAE encoding and decoding, whose reconstruction fidelity can limit the quality of the generated counterfactuals.

Our contributions. To address the drawbacks of the previous methods, our contributions are as follows:

- We introduce CounterFlow, the first CFM framework that directly conditions the vector field on both the source and target class, enabling a single model to transport examples between any pair of classes in their original pixel representation. Moreover, CounterFlow generates a counterfactual

by integrating a single (learned) ODE from source to target, with no additional VAE decoding or iterative optimization at inference.

- To enable the learning of the transport map, we introduce two complementary contributions: a latent-space nearest-neighbor pairing strategy and an auxiliary classification loss. The pairing algorithm leverages the embedding space of a pretrained VAE to associate each source example with a semantically similar target example, thereby reducing the transport distance that the flow must learn. The auxiliary classification loss further guides the flow toward transformations that induce the desired change in the classifier's prediction.
- We provide an empirical evaluation covering conditional validity, plausibility, and computational efficiency, a combination rarely reported together in prior work, demonstrating that CounterFlow matches or improves on state-of-the-art generation quality at a fraction of the computational cost of GAN- or diffusion-based alternatives.

2 Background on Counterfactual Explanations

Let $X = \mathbb{R}^d$ be the input space and $Y = \{1, \dots, K\}$ the label space. We denote by $\mathbb{M}(H)$ the set of probability distributions on a space H. We assume that there exists a true data distribution $q \in \mathbb{M}(X)$, which is unknown and from which we only have access to a learning sample. We denote by $\hat{q}$ the empirical distribution over a learning sample $S = \{\mathbf{x}^{(i)}\}_{i=1}^m \subset X$, with $S \sim q^m$, so that $\hat{q} = \mathcal{U}(S)$.[1] We also denote by q_y the conditional distribution of q given a label $y \in Y$. Let $f : X \to Y$ be the classification model whose predictions we aim to explain that associates a label $y = f(\mathbf{x})$ to each example $\mathbf{x}$. Our goal is, given $\mathbf{x}_0 \sim q_{y_0}$, to generate a counterfactual explanation of $\mathbf{x}_0$ (where $y_0 = f(\mathbf{x}_0)$), denoted by $\mathbf{x}_1$; the problem is formulated by Wachter et al. (2017) (and later reformulated by Artelt & Hammer (2019)) as

$$\mathbf{x}_1 = \text{argmin}_{\mathbf{x}'}\ d(\mathbf{x}', \mathbf{x}_0) \text{ s.t. } f(\mathbf{x}') = y_1 \neq y_0.$$

This objective makes $\mathbf{x}_1$ an adversarial example: an example close to $\mathbf{x}_0$ for the distance $d(\cdot, \cdot)$ that f classifies as y_1.

However, this definition can yield implausible examples. Indeed, according to the true data distribution q, sampling $\mathbf{x}_1$ may be unlikely, *i.e.*, $q_{y_1}(\mathbf{x}_1)$ may be small. Hence, in this paper, we instead adopt the definition of Artelt & Hammer (2020), where we aim to generate a plausible example $\mathbf{x}_1$, *i.e.*, one for which $q_{y_1}(\mathbf{x}_1)$ is higher than a constant $\varepsilon > 0$:

$$\mathbf{x}_1 = \text{argmin}_{\mathbf{x}'}\ d(\mathbf{x}', \mathbf{x}_0) \text{ s.t. } f(\mathbf{x}') = y_1 \neq y_0 \text{ and } q_{y_1}(\mathbf{x}') \geq \varepsilon. \tag{1}$$

This remains a difficult problem, as it requires both defining the distance d and solving a generally non-convex optimization problem. We therefore approximate it: to generate plausible examples $\mathbf{x}_1$, we use a conditional flow matching model, introduced in the next section. Moreover, to indirectly minimize the distance d while respecting the condition $f(\mathbf{x}_1) = y_1 \neq y_0$, we further introduce our method: CounterFlow.

3 Background on Conditional Flow Matching

Generative models aim to approximate q from $\hat{q}$ by starting from a base distribution $p_0 \in \mathbb{M}(X)$ (classically a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, with $\mathbf{0}$ the zero vector and $\mathbf{I}_d$ the $d \times d$-identity matrix) and learning to transport it toward the target distribution $p_1 \in \mathbb{M}(X)$, so that $p_1 = q$. A Continuous Normalizing Flow (CNF, Chen et al., 2018) is a generative model that achieves this by defining a time-dependent velocity field $v_\theta(\mathbf{x}, t)$ parameterized by θ, which generates samples according to

$$\frac{d\mathbf{x}_t}{dt} = v_\theta(\mathbf{x}_t, t), \quad t \in [0, 1]. \tag{2}$$

Thanks to this equation, starting from $\mathbf{x}_0 \sim p_0$, the point $\mathbf{x}_1 \sim p_1$ is obtained by solving the integrated version of Equation (2), *i.e.*, we have

$$\mathbf{x}_t = \psi_t(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^t v_\theta(\mathbf{x}_s, s) ds, \text{ where } \mathbf{x}_t \sim p_t. \tag{3}$$

[1] q^m denotes the joint distribution on a sample of size m, while $\mathcal{U}$ is a uniform distribution on a given set.

In other words, the distribution p_t is defined by the push-forward measure $p_t = (\psi_t)_{\#}p_0$, and so ψ_t transports samples from the base distribution p_0 to the distribution p_t. It defines a probability path $(p_t)_{t\in[0,1]}$ interpolating between a source distribution p_0 and a target distribution p_1. Hence, training a CNF amounts to finding θ such that p_1 approximates q, given only $\hat{q}$. However, training CNFs through maximum likelihood can be computationally expensive due to the need to solve ODEs during optimization. Besides, there exists an infinite number of probability paths, which leads to instability during training (see *e.g.*, Lipman et al., 2023). Hence, Flow Matching (FM) reformulates the training of continuous flows as a supervised regression problem on velocity fields. Instead of learning the transport dynamics indirectly through maximum likelihood, FM supervises the velocity field along predefined trajectories connecting the source and target distributions. The model is then trained to match this target velocity via the objective

$$\mathcal{L}_{\text{FM}}(\theta) = \mathop{\mathbb{E}}_{t\sim\mathcal{U}([0,1]),\mathbf{x}_t\sim p_t} \left[\|v_\theta(\mathbf{x}_t, t) - v^*(\mathbf{x}_t, t)\|^2 \right]. \tag{4}$$

However, v^* depends on the unknown q, making $\mathcal{L}_{\text{FM}}(\theta)$ intractable in practice. Fortunately, Lipman et al. (2023, Theorem 2) proved that the loss $\mathcal{L}_{\text{FM}}(\theta)$ has the same gradient as the following loss

$$\mathcal{L}'_{\text{CFM}}(\theta) = \mathop{\mathbb{E}}_{\substack{t\sim\mathcal{U}([0,1]),\\ z\sim\gamma,\mathbf{x}_t\sim p_{t,z}}} \left[\|v_\theta(\mathbf{x}_t, t) - v^*_z(\mathbf{x}_t, t)\|^2 \right],$$

where z is now a conditioning variable drawn from a coupling distribution γ that determines which distribution $p_{t,z}$ and target velocity v^*_z govern each sample. Classically (Lipman et al., 2023; Tong et al., 2024), $z = (\mathbf{x}_0, \mathbf{x}_1) \sim \gamma$ is a pair of examples, with

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \quad p_{t,z} = \delta_{\mathbf{x}_t}, \quad \text{and} \quad v^*_z(\mathbf{x}_t, t) = \tfrac{d\mathbf{x}_t}{dt} = \mathbf{x}_1 - \mathbf{x}_0. \tag{5}$$

In other words, the conditional path $p_{t,z} = \delta_{\mathbf{x}_t}$ is a Dirac point mass (*i.e.*, a degenerate distribution) supported on $\mathbf{x}_t$, and v^*_z is the (unique) velocity field consistent with $p_{t,z}$ through the continuity equation (Lipman et al., 2023). Moreover, p_0 does not need to be Gaussian nor γ to be independent (Tong et al., 2024): the same construction can transport examples between two data distributions, as we do in CounterFlow (see Section 4 further).

The above objective $\mathcal{L}'_{\text{CFM}}(\theta)$ extends naturally to conditional generation by introducing a context variable c (*e.g.*, a class label), on which both the coupling and the velocity field depend: unlike z, which only selects the path for a given pair, c is additionally fed to the network itself so that the learned velocity field can be evaluated for any desired context at inference. The context and the pair are now drawn jointly from the same coupling, $(c, z) \sim \gamma$, and we have

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathop{\mathbb{E}}_{\substack{t\sim\mathcal{U}([0,1]),\\ (c,z)\sim\gamma,\,\mathbf{x}_t\sim p_{t,z}}} \left[\|v_\theta(\mathbf{x}_t, t, c) - v^*_z(\mathbf{x}_t, t)\|^2 \right]. \tag{6}$$

Note that the path $p_{t,z}$ and target velocity v^*_z remain indexed by z alone, as in the unconditional case, only the learned field $v_\theta(\mathbf{x}_t, t, c)$ is additionally conditioned on c, which is the mechanism CounterFlow relies on to control the source and target class at inference.

4 CounterFlow

This section introduces our method, namely CounterFlow, which learns a velocity field v_θ that generates a counterfactual to approximate Equation (1). To do so, we define the loss $\mathcal{L}_{\text{CFM}}(\theta)$ derived from the coupling γ (and a minibatch version γ^n) together with the conditional path $p_{t,z}$ of Equation (5), where $(c, z) \sim \gamma$; and we end by introducing the training and inference procedures of CounterFlow.

4.1 Defining the loss

Context c and coupling γ. Since Equation (1) transports $\mathbf{x}_0$ with $f(\mathbf{x}_0) = y_0$ toward a counterfactual $\mathbf{x}_1$ with $f(\mathbf{x}_1) = y_1$, we define the context as the pair of labels $c = (y_0, y_1) \sim \mathcal{U}(Y^2)$. Then, we sample two minibatches of size n sharing this context, $B_0 \sim \mathcal{U}(S_{y_0})^n$ and $B_1 \sim \mathcal{U}(S_{y_1})^n$, where $S_y = \{\mathbf{x} \in S \mid f(\mathbf{x}) = y\}$ is the set of examples in S that f classifies as y. Finally, we pair their examples (see below) to form the set $B \in (X^2)^n$ of n pairs z. Overall, Algorithm 1

Algorithm 1: PairByEmbedding

Input: VAE encoder E

1: Sample the labels $(y_0, y_1) \sim \mathcal{U}(Y^2)$
2: Sample minibatches $B_0 \sim \mathcal{U}(S_{y_0})^n$ and $B_1 \sim \mathcal{U}(S_{y_1})^n$.
3: **for** $i = 1, \dots, n$ **do**
4: $\quad \pi(i) \leftarrow \operatorname{argmin}_{j \in \{1,\dots,n\}} \left\| E(\mathbf{x}_0^{(i)}) - E(\mathbf{x}_1^{(j)}) \right\|_2$
5: **end for**
6: $B \leftarrow \{(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(\pi(i))})\}_{i=1}^n$
7: **return** $((y_0, y_1), B)$

defines the distribution γ^n where $(c, B) \sim \gamma^n$. Moreover, when we consider only one example z from the minibatch B, we say that $(c, z) \sim \gamma$ (as in Equation (6)), which is equivalent to drawing $(c, B) \sim \gamma^n$ and sampling uniformly a single index $I \sim \mathcal{U}(\{1, \dots, n\})$ of the minibatch, *i.e.*, we have $(c, z) \sim \gamma \iff (c, B) \sim \gamma^n, z = z^{(I)}, I \sim \mathcal{U}(\{1, \dots, n\})$.

Latent-space pairing. The pairing B, created by Algorithm 1, aims to minimize a distance $d(\mathbf{x}_1, \mathbf{x}_0)$ where $\mathbf{x}_0 \in B_0 = \{\mathbf{x}_0^{(i)}\}_{i=1}^n$ and $\mathbf{x}_1 \in B_1 = \{\mathbf{x}_1^{(j)}\}_{j=1}^n$, which first requires the choice of the distance d itself. For CounterFlow, we aim for a distance d that measures whether $\mathbf{x}_0$ and $\mathbf{x}_1$ are semantically related; hence, we use the latent space of a frozen and pretrained Variational Autoencoder (VAE) and we set

$$d(\mathbf{x}_0, \mathbf{x}_1) = \|E(\mathbf{x}_0) - E(\mathbf{x}_1)\|_2, \tag{7}$$

where $E(\mathbf{x})$ denotes the latent-space embedding of the input $\mathbf{x}$. Then, to obtain the semantically related pairs, for each example $\mathbf{x}_0^{(i)} \in B_0$, where $i \in \{1, \dots, n\}$, we associate the closest example $\mathbf{x}_1^{(\pi(i))} \in B_1$ according to the distance d, where $\pi(i) = \operatorname{argmin}_{j \in \{1,\dots,n\}} d(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(j)})$ is the index in B_1 of the nearest neighbor of $\mathbf{x}_0^{(i)}$ (see Line 4).

Loss $\mathcal{L}_{\text{CFlow}}(\theta)$. We are now able to define the loss $\mathcal{L}_{\text{CFlow}}(\theta)$ minimized during the training procedure. This loss is the weighted sum $\mathcal{L}_{\text{CFlow}}(\theta) = \mathcal{L}_{\text{CFM}}(\theta) + \lambda \cdot \mathcal{L}_{\text{Cla}}(\theta)$ of a CFM-like loss $\mathcal{L}_{\text{CFM}}(\theta)$ and a loss $\mathcal{L}_{\text{Cla}}(\theta)$ dedicated to the change-of-class task.

The loss $\mathcal{L}_{\text{CFM}}(\theta)$ is obtained from Equation (6) with our coupling γ, and with the conditional path $p_{t,z}$ and the velocity $v_z^* = \mathbf{x}_1 - \mathbf{x}_0$ of Equation (5):

$$\begin{aligned}\mathcal{L}_{\text{CFM}}(\theta) &= \mathop{\mathbb{E}}_{\substack{t \sim \mathcal{U}([0,1]),\\ ((y_0,y_1),(\mathbf{x}_0,\mathbf{x}_1)) \sim \gamma}} \|v_\theta(\mathbf{x}_t, t, y_0, y_1) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2 \\ &= \mathop{\mathbb{E}}_{\substack{t \sim \mathcal{U}([0,1]),\\ ((y_0,y_1),B) \sim \gamma^n}} \sum_{(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}) \in B} \tfrac{1}{n} \|v_\theta(\mathbf{x}_t^{(i)}, t, y_0, y_1) - (\mathbf{x}_1^{(i)} - \mathbf{x}_0^{(i)})\|^2.\end{aligned}$$

Moreover, while Equation (1) requires generating realistic examples (according to q_{y_1}) with a minimal distance d, it also requires an example $\mathbf{x}_1$ such that $f(\mathbf{x}_1) = y_1$. We add the loss $\mathcal{L}_{\text{Cla}}(\theta)$, weighted by $\lambda > 0$, to further enforce a velocity field leading in the right direction. It is defined by

$$\mathcal{L}_{\text{Cla}}(\theta) = \mathop{\mathbb{E}}_{\substack{t \sim \mathcal{U}([0,1]),\\ ((y_0,y_1),(\mathbf{x}_0,\mathbf{x}_1)) \sim \gamma}} [t\, \text{CE}(f(\mathbf{x}_{t+\delta}), y_1)] = \mathop{\mathbb{E}}_{\substack{t \sim \mathcal{U}([0,1]),\\ ((y_0,y_1),B) \sim \gamma^n}} \sum_{(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}) \in B} \tfrac{1}{n} \left[t\, \text{CE}\Big(f(\mathbf{x}_{t+\delta}^{(i)}), y_1 \Big) \right],$$

where CE is the classical multiclass cross-entropy loss, and $\mathbf{x}_{t+\delta}$ is defined as taking a small Euler step along the predicted velocity, *i.e.* $\mathbf{x}_{t+\delta} = \mathbf{x}_t + \delta\, v_\theta(\mathbf{x}_t, t, y_0, y_1)$. The weighting factor t progressively increases the influence of the classification objective: close to the source, the optimization is dominated by the flow matching objective, while near the target the classification constraint becomes stronger.

4.2 Training procedure (Algorithm 2)

To minimize the loss function $\mathcal{L}_{\text{CFlow}}(\theta)$, we use the training loop described in Algorithm 2. A training step can be described in the following way. We first sample $((y_0, y_1), B) \sim \gamma^n$ thanks to

Algorithm 2: Training procedure of CounterFlow

Input: Source distribution p_0, target distribution p_1, flow network v_θ, VAE encoder E, classifier f

1: **while** Training **do**
2: Sample $((y_0, y_1), B) \leftarrow \text{PairByEmbedding}(E)$.
3: **for** $(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}) \in B$ **do**
4: Sample $t \sim \mathcal{U}(0, 1)$.
5: Compute $\mathbf{x}_t^{(i)} \leftarrow (1-t)\mathbf{x}_0^{(i)} + t\mathbf{x}_1^{(i)}$ and $\mathbf{x}_{t+\delta}^{(i)} \leftarrow \mathbf{x}_t^{(i)} + \delta\, v_\theta(\mathbf{x}_t^{(i)}, t, y_0, y_1)$.
6: Compute the flow matching loss $\mathcal{L}_{\text{CFM}}^{(i)}(\theta) \leftarrow \left\| v_\theta(\mathbf{x}_t^{(i)}, t, y_0, y_1) - (\mathbf{x}_1^{(i)} - \mathbf{x}_0^{(i)}) \right\|_2^2$.
7: Compute the classification loss $\mathcal{L}_{\text{Cla}}^{(i)}(\theta) \leftarrow t\,\text{CE}\Big(f(\mathbf{x}_{t+\delta}^{(i)}), y_1\Big)$
8: **end for**
9: Update θ using $\frac{1}{n}\sum_{i=1}^{n}\Big[\mathcal{L}_{\text{CFM}}^{(i)}(\theta) + \lambda \cdot \mathcal{L}_{\text{Cla}}^{(i)}(\theta)\Big]$.
10: **end while**
11: **return** v_θ

Algorithm 1 (line 2), which corresponds to the labels (y_0, y_1) and the minibatch B with the pairs of examples established using nearest-neighbor matching in the latent space of a pretrained VAE encoder. Then, for each matched pair $(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}) \in B$, we sample $t \sim \mathcal{U}([0, 1])$ (line 4) and construct the interpolated samples (line 5). Then, we compute the losses for $(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}) \in B$ in lines 6 and 7. After computing the loss on the minibatch, we update v_θ in line 9 using, *e.g.*, Adam optimizer.

4.3 Inference

Once v_θ is trained, generating a counterfactual no longer requires any pairing, VAE encoding, or minibatch construction. CounterFlow requires only an input sample $\mathbf{x}_0$ (with the prediction $y_0 = f(\mathbf{x}_0)$) and a desired target class y_1. The source class is obtained using the pretrained classifier, while the target class is either specified by the user or sampled uniformly among the remaining classes. Starting from the input $\mathbf{x}_0$, we solve the learned ordinary differential equation defined in Equation (2), by integrating from $t = 0$ to $t = 1$, as shown in Equation (3). The endpoint of the trajectory is returned as the generated counterfactual explanation $\mathbf{x}_1$.

5 Experiments

We now turn to an empirical evaluation of CounterFlow against state-of-the-art counterfactual generation methods. Note that our implementation is based on the `torchcfm` library (Tong et al., 2024), which provides the Conditional Flow Matching framework used to train CounterFlow. The source code to reproduce the experiments will be made publicly available at this link.

5.1 Setting

Datasets. We evaluate CounterFlow on two standard image classification benchmarks: **MNIST** and **CIFAR-10**, MNIST consists of 70,000 grayscale images of handwritten digits belonging to 10 classes, with a spatial resolution of 28×28 pixels. CIFAR-10 contains 60,000 color images of size 32×32 pixels, evenly distributed among 10 object categories. These two datasets provide complementary evaluation settings, ranging from a simple handwritten digit classification task to a natural image classification benchmark. For all experiments, we use the standard training and test splits provided with each dataset. Section A.4 discusses why we do not extend the comparison to larger benchmarks such as ImageNet.

CounterFlow's inference. For all quantitative evaluations, CounterFlow is integrated via the Euler solver with 40 steps. This fixed-step solver provides a more deterministic evaluation protocol and allows a fair comparison of the computational cost across methods. For qualitative visualizations, we instead use the Dormand–Prince solver (Dormand & Prince, 1980) (`dopri5`) to obtain smoother trajectories and higher-quality visual results.

Methods. We compare CounterFlow[2] against three recent counterfactual generation methods: C3LT (Khorram & Fuxin, 2022), FastDiME (Weng et al., 2024), and LeapFactual (Cao et al., 2025), where we use the official implementations. Among diffusion-guidance methods, we retain only FastDiME as a representative baseline: DiME and ACE are prohibitively expensive to run at scale due to their inner-loop or per-step attack computations; DVCE requires an adversarially robust classifier, a constraint that our method does not rely on and that we therefore do not impose in our experiments; and MaskDiME's masking mechanism restricts edits to a localized region, which conflicts with our evaluation setting where the change may need to affect the whole image. Beyond this choice, neither of the two remaining baselines, namely LeapFactual and FastDiME, is applicable to both datasets. LeapFactual relies on a compact VAE latent space to remain tractable, which does not scale to CIFAR-10 without dominant reconstruction artifacts and a much costlier latent flow, so we evaluate it only on MNIST. Conversely, FastDiME depends on pretrained OpenAI diffusion models built for natural images, so we evaluate it only on CIFAR-10.

Models to be explained. To ensure a fair comparison, all methods rely on the same pretrained image classifier for a given dataset. For MNIST, we trained a simple convolutional neural network achieving an accuracy of 99.9% on the test set. For CIFAR-10, we use a pretrained RepVGG-A2 classifier (Ding et al., 2021) with publicly available weights from the `pytorch-cifar-models` repository (distributed through the official PyTorch Hub interface), achieving a test accuracy of 94.8%. Both classifiers are frozen during training of counterfactual models and are used consistently for counterfactual generation and metrics computation across all competing methods.

Metrics. We evaluate the counterfactual explanations along four axes: *effectiveness* (reaching $f(\mathbf{x}_1) = y_1$), *proximity to the source* (a distribution-level counterpart of the distance $d(\mathbf{x}_1, \mathbf{x}_0)$ minimized per example in Equation (1), measured in a semantic feature space rather than in a pixel space), *realism* (the plausibility constraint $q_{y_1}(\mathbf{x}_1) \geq \varepsilon$, also from Equation (1)), and *computational efficiency*. We operationalize the evaluation of these axes through the following five metrics.

- **Conditional Validity** measures the proportion (in %) of generated counterfactuals x_1 that are classified as the desired target class y_1, *i.e.*, whether we have $f(\mathbf{x}_1) = y_1$.
- **FID**$_\mathrm{t}$ computes the Fréchet Inception Distance between the generated counterfactuals and real samples from the target class, quantifying *realism*: lower values indicate that the generated samples better match the distribution of the desired class.
- **FID**$_\mathrm{s}$ computes the Fréchet Inception Distance between the generated counterfactuals and the original source class, measuring *proximity to the source*: lower values indicate that the counterfactuals remain closer to the original data distribution, suggesting that fewer semantic changes were applied to reach the target class.
- **FLOPs / sample** measures the computational cost required to generate a single counterfactual explanation. It includes the complete inference pipeline, *i.e.*, all neural network evaluations required by the ODE solver. This metric provides an implementation-independent estimate of inference complexity.
- **Model Size** reports the total number of trainable parameters of each method, allowing comparison of their memory footprint and model complexity.

Following the standard definition of the Fréchet Inception Distance (FID, Heusel et al., 2017), feature statistics for FID$_\mathrm{t}$ and FID$_\mathrm{s}$ are extracted from pretrained image classification networks. For CIFAR-10, we use the pretrained Inception-V3 network available in `torchvision`. Since ImageNet-pretrained models are not well suited for handwritten digit recognition, for MNIST we instead compute features using the pretrained convolutional classifier employed throughout our experiments. FID is then computed between the Gaussian statistics of the extracted features from the generated counterfactuals and those of the corresponding source or target class, *i.e.*, FID$_\mathrm{t}$ on the target class and FID$_\mathrm{s}$ on the source class. Note that, since these statistics are estimated from the subset of test examples belonging to a single class, rather than from the full test set as in standard, unconditional FID computations, the reported values are higher than those typically observed in unconditional image generation. We therefore focus on relative comparisons between methods under the same evaluation protocol rather than on the absolute reported numbers.

[2]The hyperparameters of CounterFlow are listed in Appendix.

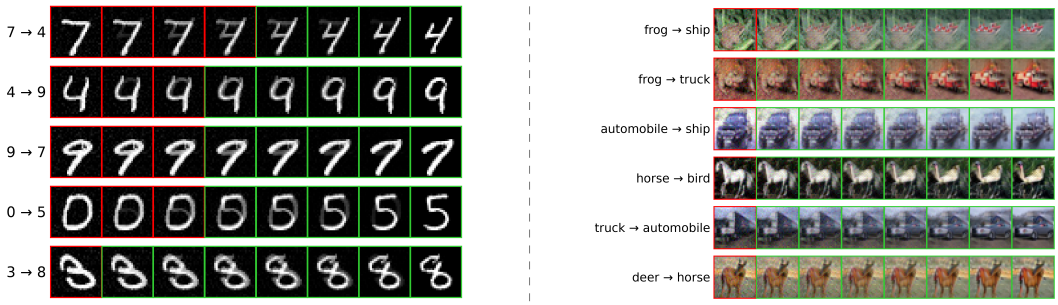
Table 1: (a) Quantitative results and (b) Ablation of the pairing used to train CounterFlow: latent-space nearest neighbor (Algorithm 1) vs. random. Higher conditional validity (CV) is better, while lower FID scores indicate that the generated counterfactuals are closer to the target or source distributions. Lower FLOPs/sample and model size (MS) are preferred for computational efficiency.

Method	CV↑	FID_t↓	FID_s↓	FLOPs↓	MS↓
MNIST					
C3LT	99.0	187.89	4310.01	223.86M	480.81M
LeapFactual	98.6	217.08	**2625.52**	**0.86M**	**56.95K**
CounterFlow	**99.6**	**105.70**	3224.54	174.66M	4.64M
CIFAR10					
C3LT	99.0	115.04	153.41	3.93G	481.33M
FastDiME	16.0	159.86	175.64	1586.63G	52.55M
CounterFlow	**100.0**	**62.54**	**117.03**	**1.73G**	**39.36M**

(a) Quantitative results

Pairing	CV↑	FID_t↓	FID_s↓
MNIST			
Random	99.4	111.98	3634.20
Latent	**99.6**	**105.70**	**3224.54**
CIFAR10			
Random	99.9	63.27	125.07
Latent	**100.0**	**62.54**	**117.03**

(b) Pairing used to train CounterFlow



(a) MNIST (b) CIFAR10

Figure 2: CounterFlow trajectories for five (MNIST) and six (CIFAR10) samples, from source to target class (labeled for CIFAR10). Border color indicates the classifier's prediction along the path: red for the source class, green for the target class.

5.2 Quantitative Results

Table 1a reports the quantitative comparison between CounterFlow and representative counterfactual generation methods on MNIST and CIFAR-10; a radar-plot visualization is provided in Figure 3 in the appendix.

MNIST. CounterFlow reaches the highest conditional validity (99.6%) and the lowest target FID while remaining close to the source, confirming that the learned conditional transport reliably reaches the target class with a single ODE integration. LeapFactual stays closer to the source but has the highest target FID, reflecting a more conservative transformation. C3LT performs worst overall and requires by far the largest model, due to its per-class-pair VAE, GAN, and mapping networks.

CIFAR-10. CounterFlow is best on all five metrics: it reaches 100% conditional validity, nearly halves C3LT's target FID while also lowering its source FID by a quarter, and needs roughly half the inference FLOPs and an order of magnitude fewer parameters. Improving both FIDs at once is the signature of a near-minimal transport: counterfactuals that fall within the target class without drifting away from the source image. FastDiME reaches only 16% conditional validity despite a diffusion backbone 900× more expensive per sample, and has the worst FID on both axes; with so few samples switching class, these scores reflect generic realism rather than class transfer.

Overall, CounterFlow achieves state-of-the-art counterfactual validity and the best realism/proximity trade-off on CIFAR-10, while substantially reducing inference complexity compared with prior flow- and diffusion-based approaches.

Ablation. Table 1b shows that replacing latent-space pairing with random pairing consistently lowers conditional validity and raises both target and source FID on both datasets, mostly through the source FID on CIFAR-10. Pairing semantically similar examples yields simpler transport trajectories that are easier for the conditional velocity field to approximate, whereas random pairing often associates visually unrelated samples, leading to harder-to-learn trajectories and lower-quality counterfactuals. Section A.3 further ablates λ, the conditioning context and the explained classifier.

5.3 Qualitative Results

Style Preservation. Figure 1 shows five source/counterfactual pairs per dataset. On MNIST, CounterFlow preserves writer-specific style, such as stroke thickness and slant – *e.g.*, the $4 \rightarrow 5$ transition retains the source's thin strokes – and keeps the digit's orientation, rather than collapsing to a generic class prototype. On CIFAR10, the same near-minimal-path behavior appears at the scene level: the background stays untouched and only the foreground object changes, often preserving its color palette and orientation while altering just the class-relevant structure.

Trajectories. Figure 2 visualizes the full integration trajectory for five counterfactuals per dataset, with the border color marking the classifier's prediction at each step (red for the source class, green for the target class). In most examples the color flips early, showing that CounterFlow reaches the target class within a small fraction of the path and spends the remaining steps refining the sample within that class rather than driving the decision itself, *i.e.*, most of the trajectory makes the class change plausible rather than induces it. Intermediate frames stay visually close to both endpoints rather than passing through a degraded intermediate state, confirming at a finer granularity that CounterFlow produces minimal, semantically meaningful edits rather than an unconstrained image-to-image mapping. See Figures 4 and 5 in Section A.5 for additional CounterFlow trajectories.

6 Discussion

Existing image counterfactual generation methods often require heavy generative backbones, including diffusion models or multiple class-specific generative networks, making their deployment computationally demanding. CounterFlow demonstrates that Conditional Flow Matching provides a significantly simpler alternative. By learning a single target-conditioned transport field and requiring only one ODE integration at inference, our method achieves competitive or superior counterfactual quality while substantially reducing computational complexity. We believe this constitutes an important step toward making counterfactual explanations practical for interactive decision-support systems, where explanations must be generated quickly and consistently.

Despite these encouraging results, several limitations remain. First, our experiments are restricted to relatively low-resolution image datasets. Evaluating CounterFlow on larger-scale benchmarks such as ImageNet or high-resolution generative datasets would provide further evidence of its scalability. Second, our latent-space pairing strategy currently relies on nearest-neighbor retrieval using a pretrained encoder, which introduces an additional preprocessing step. Finally, while Conditional Flow Matching substantially reduces inference complexity compared to diffusion models, solving an ODE remains more expensive than a single feed-forward network.

7 Conclusion

We presented CounterFlow, a Conditional Flow Matching framework for visual counterfactual explanation generation. Unlike previous approaches relying on GANs or diffusion models, CounterFlow directly learns a target-conditioned transport field between image classes in pixel space. Experiments on MNIST and CIFAR-10 demonstrate that the proposed method achieves state-of-the-art counterfactual validity while remaining competitive in image realism and substantially reducing computational requirements.

Beyond counterfactual explanations, CounterFlow illustrates that target-conditioned flow matching can efficiently learn semantic transformations directly between complex data distributions, without relying on an intermediate noise distribution. Future work will investigate its application to larger-scale datasets and other data transformation tasks, including domain adaptation, semantic image editing, and robust model analysis.

AI USE STATEMENT

We used generative AI tools (LLM-based coding assistants) to help implement and debug parts of the experimental code, and to polish the phrasing of some sentences in this manuscript. We did not use generative AI tools to generate research ideas, to design the method, or to produce or interpret experimental results. All AI-assisted code and text were reviewed, tested, and validated by the authors, who take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work introduces a method for generating visual counterfactual explanations of image classifiers, evaluated on the standard MNIST and CIFAR-10 benchmarks, which do not contain personal or sensitive data. As with other counterfactual-explanation and image-editing methods, the underlying generative technology could in principle be misused to produce misleading edited images of real people; we mitigate this risk by restricting our experiments to non-human-identifying benchmark datasets and by releasing the method as a tool for model interpretability rather than for unconstrained image generation.

REPRODUCIBILITY STATEMENT

We describe the CounterFlow training and inference procedures in full in Section 4, including the pairing algorithm (Algorithm 1) and the training loop (Algorithm 2). All training hyperparameters for both datasets are listed in Table 2 in the appendix. We use only the standard, publicly available MNIST and CIFAR-10 datasets and publicly available pretrained classifiers (Section 5). Our implementation builds on the open-source `torchcfm` library, and we will release the source code needed to reproduce all experiments upon publication.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *International Conference on Learning Representations*, 2023.

André Artelt and Barbara Hammer. On the computation of counterfactual explanations - A survey. *arXiv*, abs/1911.07749, 2019.

André Artelt and Barbara Hammer. Convex Density Constraints for Computing Plausible Counterfactual Explanations. In *International Conference on Artificial Neural Networks*, 2020.

Maximilian Augustin, Valentyn Boreiko, Francesco Croce, and Matthias Hein. Diffusion Visual Counterfactual Explanations. In *Advances in Neural Information Processing Systems*, 2022.

Zhuo Cao, Xuan Zhao, Lena Krieger, Hanno Scharr, and Ira Assent. LeapFactual: Reliable Visual Counterfactual Explanation Using Conditional Flow Matching. In *Advances in Neural Information Processing Systems*, 2025.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, 2018.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

J. R. Dormand and P. J. Prince. A Family of Embedded Runge-Kutta Formulae. *Journal of Computational and Applied Mathematics*, 1980.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (XAI): Core Ideas, Techniques, and Solutions. *ACM Computing Surveys*, 2023.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A. Mann. Improving Robustness using Generated Data. In *Advances in Neural Information Processing Systems*, 2021.

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual Visual Explanations. In *International Conference on Machine Learning*, 2019.

Changlu Guo, Anders Nymark Christensen, Anders Bjorholm Dahl, and Morten Rieger Hannemose. MaskDiME: Adaptive Masked Diffusion for Precise and Efficient Visual Counterfactual Explanations. *arXiv*, abs/2602.18792, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. Adversarial Counterfactual Visual Explanations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. Diffusion Models for Counterfactual Explanations. *Computer Vision and Image Understanding*, 2024.

Mark T Keane and Barry Smyth. Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI). In *International Conference on Case-Based Reasoning*, 2020.

Saeed Khorram and Li Fuxin. Cycle-Consistent Counterfactuals by Latent Transformations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. AudioLDM: Text-to-Audio Generation with Latent Diffusion Models. In *International Conference on Machine Learning*, 2023a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations*, 2023b.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *arXiv*, abs/1711.00399, 2017.

Nina Weng, Paraskevas Pegios, Eike Petersen, Aasa Feragen, and Siavash Arjomand Bigdeli. Fast Diffusion-Based Counterfactuals for Shortcut Removal and Generation. In *European Conference on Computer Vision*, 2024.