Back to the Feature: Explaining Video Classifiers with Video Counterfactual Explanations

Supplementary Material

A. Diffusion Model Background

A.1. Denoising Diffusion Probabilistic Models

Denoising diffusion probabilistic models (DDPMs) [5, 12, 13] consist of two Markov processes: a forward process that gradually adds Gaussian noise to a clean sample $\mathbf{x}_0 \in \mathbb{R}^{n\times c\times h\times w}$ until obtaining a highly-noised sample $\mathbf{x}_T \in \mathbb{R}^{n\times c\times h\times w}$, and a reverse process that removes the noise on x_T to recover the original data distribution. The forward process is defined as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)I), \tag{1}$$

where $\alpha_t \in \{\alpha_t : 1 \leq t \leq T\}$ defines the noise schedule, I is the identity matrix, and n, c, h and w denote the number, channel, height and width of frames, respectively. Alternatively, given $\mathbf{x}_0$, the noised sample $\mathbf{x}_t$ at any timestep t can be analytically derived from:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{2}$$

where $\bar{\alpha}_t := \prod_{i=1}^{t}\alpha_i$ and $\epsilon \in \mathbb{R}^{n\times c\times h\times w}$ is the Gaussian noise sampled from the standard Gaussian distribution. In the reverse process, a denoiser (e.g., U-Net or Diffusion Transformer) parameterized by ϕ is trained to reverse the forward process:

$$p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mu_\phi(\mathbf{x}_t, t), \Sigma_\phi(\mathbf{x}_t, t)), \tag{3}$$

where μ_ϕ and Σ_ϕ are the predicted mean and variance of the data distribution of $\mathbf{x}_t$, respectively. The less-noisy sample $\mathbf{x}_{t-1}$ is drawn from $p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ step by step until the noise-free sample x_0 is obtained. In most cases, a denoiser estimates the mean μ_ϕ by predicting the added noise ϵ_ϕ between $\mathbf{x}_t$ and $\mathbf{x}_0$:

$$\mu_\phi(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\phi(\mathbf{x}_t, t)\right) \tag{4}$$

A.2. Denoising Diffusion Implicit Models

In order to speed up the generation process at the inference time, $\mathbf{x}_{t-1}$ can be calculated in a deterministic way using the denoising diffusion implicit model (DDIM) [11] sampler:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\phi(\mathbf{x}_t, t) \tag{5}$$

where $\hat{\mathbf{x}}_0$ is the clean sample approximated at the timestep t by rewriting Eq. (2):

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\phi(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}} \tag{6}$$

A.3. Classifier-guidance

To condition the class of generated samples, Dhariwal et al. [3] proposed classifier guidance, where an external classifier f_θ is introduced into the denoising process of diffusion models. Specifically, the classifier's gradients with respect to noised images $\nabla_{\mathbf{x}_t}\log f_\theta(\mathbf{y}_c \mid \mathbf{x}_t)$ steer the denoising trajectory toward the data manifold of the target class $\mathbf{y}_c \in \mathbb{R}^k$ by modifying the predictions of the denoiser:

$$\hat{\epsilon}_\phi(\mathbf{x}_t, t, \mathbf{y}_c) = \epsilon_\phi(\mathbf{x}_t, t) - w\sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t}\log f_\theta(\mathbf{y}_c \mid \mathbf{x}_t) \tag{7}$$

where $\hat{\epsilon}_\phi(\mathbf{x}_t, t, \mathbf{y}_c)$ is the modified prediction of the denoiser and w is the guidance scale.

B. Method Details

The optimization framework of BTTF is summarized in Algorithm 1. We define "denoising loop" as a flow matching sampling process where the noise-free latent $\hat{\mathbf{z}}_0$ is obtained by recursively running Eq. (1) n times, as summarized in Algorithm 2. The whole framework consists of two stages: inversion and CFE generation. In the first stage, the sampled latent $\mathbf{z}_T$ is optimized to be close to the latent representation of the original input video $\mathbf{z}_i$. In the second stage, we progressively optimize the initial latent $\mathbf{z}_T$ to search for a counterfactual video $\hat{\mathbf{x}}_c$ that changes the prediction of the target classifier f_θ towards the target class $\mathbf{y}_c$. In addition, gradient checkpointing is utilized to mitigate the GPU memory consumption of the proposed method. The hyperparameters of CFE generation experiments for different target classifiers are shown in Tab. S1. Notably, for the target classifier A-swinR, because the semantic differences among action classes in the NTU RGB+D training set may be relatively large (e.g., single-person "salute" vs. two-person "hugging"), the number of inversion iterations, K_I, needs to be manually adjusted according to the gap between the class of the original input video and the target class. This empirical heuristic suggests using a smaller K_I when the semantic gap between the original and target classes is large, and a larger K_I otherwise.

C. Details of Model Training

C.1. Video Dataset Preprocessing

Shape-Moving. The Shape-Moving dataset is our synthetic video dataset consisting of 64k videos. Each video contains 65 frames with a spatial resolution of $3\times224\times224$ (channel $\times$ width $\times$ height) and a frame rate of 16 fps. The dataset

Table S1. **Hyperparameters used in the BTTF optimization algorithm.**

Hyperparameter	M-swin	E-swin	A-swinR
# Inversion iterations K_I	0	40	0~40
# CFE generation iterations K_C	100	100	100
Style loss regularization coefficient λ	1×10^5	1×10^5	1×10^5
Max number of denoising steps N	15	15	15

Table S2. **Training configurations for video classifiers.**

Model	Training Dataset	Input Clip Shape	# Training Epochs
M-swin	Shape-Moving	$32 \times 3 \times 224 \times 224$	1
E-swin	MEAD	$16 \times 3 \times 512 \times 512$	15
A-swinR	NTU RGB+D	$16 \times 3 \times 448 \times 448$	30

includes four directional categories: "Up", "Down", "Left", and "Right", with 16k videos per class. We randomly split the dataset into training and testing sets using a 4:1 ratio.

MEAD. [16] The raw MEAD videos have a spatial resolution of $3 \times 1080 \times 1920$. For preprocessing, we first resize the shorter spatial dimension (height) to 512 while preserving the aspect ratio, which is followed by a center crop along the longer dimension to obtain videos with a spatial resolution of 512×512. In the temporal domain, we perform either downsampling or upsampling so that all videos are normalized to 33 frames, which are then saved at 8 fps, resulting in clips of approximately 4 seconds. The dataset is randomly divided into training and testing sets with a 4:1 split.

NTU RGB+D. [9] The preprocessing pipeline for NTU RGB+D follows the same procedure as that used for the MEAD dataset, including spatial resizing and center cropping to 512×512, as well as temporal normalization to 33 frames at 8 fps.

C.2. Video Classifiers Training

We trained[1] three target video classifiers, namely M-swin, E-swin, and A-swinR, which are built upon an ImageNet-1K pretrained Video Swin Transformer [8] backbone. The training sets, input clip shape ($n \times c \times h \times w$), and number of training epochs, are summarized in Tab. S2. The input clips are uniformly sampled from videos in the training set using a temporal stride of 2.

D. Evaluation Details

D.1. Metrics

Flip Rate (FR) measures the effectiveness of counterfactual examples in changing the classifier's prediction to the target class. Given a counterfactual video x_c, a target classifier f_θ, and a target class y_c, FR is defined as:

$$\mathrm{FR}(\mathbf{x}_c, \mathbf{y}_c) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\big[f_\theta(\mathbf{x}_c) = \mathbf{y}_c\big], \tag{8}$$

where $\mathbb{I}$ denotes the indicator function and M is the total number of samples. A higher FR indicates more effective counterfactual interventions.

SSIM [17] quantifies perceptual similarity between two corresponding frames in input and counterfactual videos based on luminance, contrast, and structural information.

[1] Code adapted from https://github.com/open-mmlab/mmaction2

Algorithm 1 BTTF optimization algorithm

Input: Original video $\mathbf{x}_i$, target class y_c, target video classifier f_θ, max number of denoising steps N, number of inversion iterations K_I, number of CFE generation iterations K_C, the objective functions $\mathcal{L}_I$ and $\mathcal{L}_C$
Output: CFE video $\mathbf{x}_c$

$\mathbf{I} \leftarrow$ first_frame($\mathbf{x}_i$) ▷ Extract the original first frame
$\mathbf{I}_C \leftarrow$ CLIP($\mathbf{I}$) ▷ Extract CLIP embedding
$\mathbf{I}_p \leftarrow$ zero_pad_to_video($\mathbf{I}$) ▷ Zero-pad the first frame
$\mathbf{I}' \leftarrow \mathcal{E}(\mathbf{I}_p)$ ▷ Encode the zero-padded first frame
$\mathbf{z}_i \leftarrow \mathcal{E}(\mathbf{x}_i)$ ▷ Encode the original video
$\mathbf{z}_T \leftarrow$ sample_normal_like($\mathbf{z}_i$) ▷ Sample the latent
optim $\leftarrow$ AdamW($\mathbf{z}_T$) ▷ Initialize the optimizer
▷ **Stage 1: Inversion**
$n \leftarrow 1$
for $k \leftarrow 1, \ldots, K_I$ **do**
  optim.zero_grad()
  $\hat{\mathbf{z}}_0 \leftarrow$ denoising_loop($\mathbf{z}_T, \mathbf{I}', \mathbf{I}_C, n$)
  $l \leftarrow \mathcal{L}_I(\hat{\mathbf{z}}_0, \mathbf{z}_i)$
  l.backward()
  optim.step() ▷ Update $\mathbf{z}_T$
end for
▷ **Stage 2: CFE generation**
for $n \leftarrow 1, \ldots, N$ **do**
  for $k \leftarrow 1, \ldots, K_C$ **do**
    optim.zero_grad()
    $\hat{\mathbf{z}}_0 \leftarrow$ denoising_loop($\mathbf{z}_T, \mathbf{I}', \mathbf{I}_C, n$)
    $\hat{\mathbf{x}}_c \leftarrow \mathcal{D}(\hat{\mathbf{z}}_0)$
    $\hat{y} \leftarrow f_\theta(\hat{\mathbf{x}}_c)$
    $l \leftarrow \mathcal{L}_C(\mathbf{x}_i, \hat{\mathbf{x}}_c, y_c, \hat{y})$
    l.backward()
    optim.step() ▷ Update $\mathbf{z}_T$
  end for
end for

$\mathbf{x}_c \leftarrow \mathcal{D}$(denoising_loop($\mathbf{z}_T, \mathbf{I}', \mathbf{I}_C, N$))
return $\mathbf{x}_c$

Algorithm 2 Denoising loop used in Algorithm 1

Input: Initial latent $\mathbf{z}_T$, encoded first frame $\mathbf{I}'$, CLIP embedding $\mathbf{I}_C$, number of denoising steps n
Output: Denoised latent $\hat{\mathbf{z}}_0$

$\mathbf{z} \leftarrow \mathbf{z}_T$
for $t \leftarrow n, \dots, 1$ **do**
  $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon}_\phi(\mathbf{z}, t, \mathbf{I}', \mathbf{I}_C)$
  $\mathbf{z} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \frac{\mathbf{z} - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}}{\sqrt{\bar{\alpha}_t}} + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}$
end for
$\hat{\mathbf{z}}_0 \leftarrow \mathbf{z}$
return $\hat{\mathbf{z}}_0$

For two frames $\mathbf{x}_{i,n}$ and $\mathbf{x}_{c,n}$, SSIM is computed as:

$$\mathrm{SSIM}(\mathbf{x}_{i,n}, \mathbf{x}_{c,n}) = \frac{(2\mu_{i,n}\mu_{c,n} + C_1)(2\sigma_{ic,n} + C_2)}{(\mu_{i,n}^2 + \mu_{c,n}^2 + C_1)(\sigma_{i,n}^2 + \sigma_{c,n}^2 + C_2)}, \tag{9}$$

where $\mu_{i,n}$, $\mu_{c,n}$ are means, $\sigma_{i,n}^2$, $\sigma_{c,n}^2$ are variances, $\sigma_{ic,n}$ is the covariance, and the constants C_1 and C_2 are stabilizing terms introduced to avoid numerical instability when the denominators are close to zero. Here, SSIM is averaged across frames of input and counterfactual videos: $\mathrm{SSIM}(\mathbf{x}_i, \mathbf{x}_c) = \frac{1}{N_f}\sum_{n=1}^{N_f} \mathrm{SSIM}(\mathbf{x}_{i,n}, \mathbf{x}_{c,n})$, where N_f is the number of video frames. Higher values indicate higher perceptual similarity.

LPIPS [19] measures perceptual similarity using deep network features. Here, we use AlexNet [7]. Given two images, LPIPS is defined as:

$$\mathrm{LPIPS}(\mathbf{x}_{i,n}, \mathbf{x}_{c,n}) = \sum_l w_l \left\| \tilde{f}_{A,l}(\mathbf{x}_{i,n}) - \tilde{f}_{A,l}(\mathbf{x}_{c,n}) \right\|_2^2, \tag{10}$$

where $f_{A,l}$ denotes features extracted at layer l of AlexNet, $\tilde{f}_{A,l}$ is the normalized feature vector, and w_l are learned weights. Here, LPIPS is averaged across frames of input and counterfactual videos: $\mathrm{LPIPS}(\mathbf{x}_i, \mathbf{x}_c) = \frac{1}{N_f}\sum_{n=1}^{N_f} \mathrm{LPIPS}(\mathbf{x}_{i,n}, \mathbf{x}_{c,n})$. Lower values indicate higher perceptual similarity.

FID [4] measures the discrepancy between real and generated images by comparing their Inception-V3 feature distributions. Let $f_I(\mathbf{x})$ denote the feature vector extracted from the Inception-V3 network [14]. We assume that the feature vectors of real images and generated images follow Gaussian distributions $\mathcal{N}(\mu_r, \Sigma_r)$ and $\mathcal{N}(\mu_g, \Sigma_g)$, respectively. Under this assumption, FID corresponds to the squared 2-Wasserstein distance between these two Gaussian distributions, given by

$$W_2^2\big(\mathcal{N}(\mu_r, \Sigma_r), \mathcal{N}(\mu_g, \Sigma_g)\big) = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\Big), \tag{11}$$

where Tr denotes the trace operator (the sum of the diagonal elements of a matrix). Lower values indicate that the generated images more closely match the real data distribution.

FVD [15] extends FID from images to videos by applying Eq. (11) to spatiotemporal features extracted with an I3D network [2]. As in FID, the real and generated video features are modeled as Gaussian distributions, and FVD evaluates the discrepancy between them. Lower values indicate more realistic and temporally coherent video generation.

D.2. Comparison with CG-based methods

To demonstrate the limitations of applying image-based CFE methods to videos, we adapt the classifier guidance (CG) strategy, commonly used in image-based CFE methods [1, 6, 10, 18],to the video setting and compare it with our proposed BTTF. We construct three CG baselines that progressively incorporate more temporal information: CG-Frame, which edits each frame independently; CG-Video-Mid, which applies video-classifier gradients but starts denoising from a half-noisy state; and CG-Video, which performs full-noise initialization and video-level guidance.

As shown in Tab. S3, all CG variants struggle to generate high-quality video counterfactuals. CG-Frame fails to maintain temporal consistency, resulting in extremely poor FVD despite reasonable frame-level SSIM. CG-Video-Mid partially improves temporal coherence by using video-level gradients, but the partial-noise initialization limits its ability to introduce coherent spatiotemporal edits. CG-Video, the strongest CG baseline, shows better temporal alignment, yet its FID and LPIPS remain far worse than ours, indicating that simply extending image-based classifier-guidance to videos cannot reliably edit spatiotemporal features.

In contrast, BTTF clearly outperforms CG variants across most metrics, achieving by far the lowest FID (27.49) and FVD (250.11). These results show that BTTF generates counterfactual videos that are simultaneously realistic, temporally stable, and semantically meaningful.

Overall, this comparison confirms that video counterfactual explanations cannot be obtained by naively adapting image-based classifier-guidance methods, and that our BTTF framework is specifically effective at producing coherent, high-fidelity spatiotemporal counterfactuals.

E. Qualitative results

More generated video CFEs by BTTF are shown in Fig. S1 and Fig. S2.

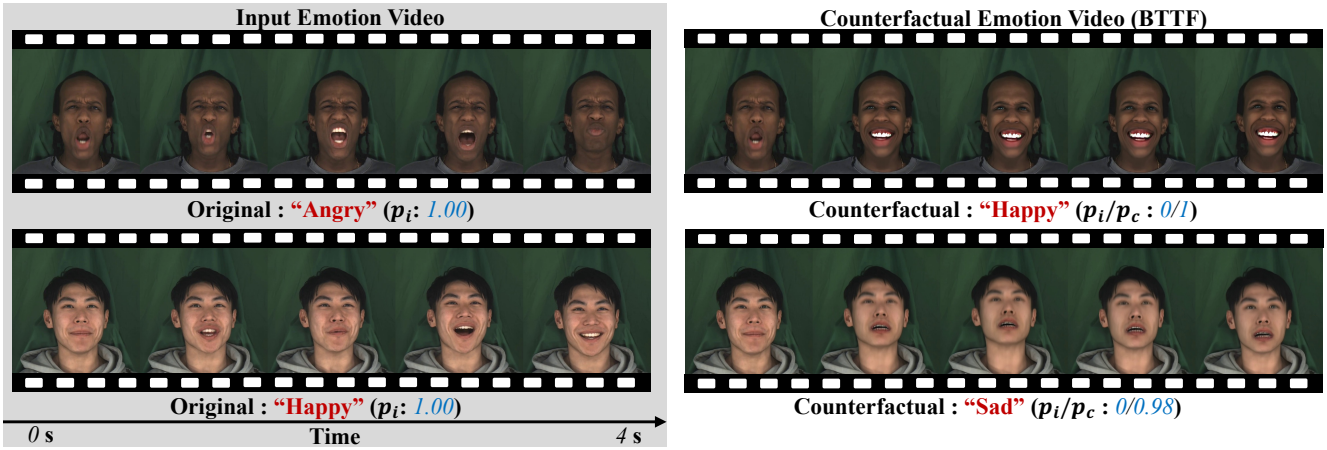

Figure S1. **Video CFEs generated by BTTF for the target video classifier E-swin trained on MEAD.**

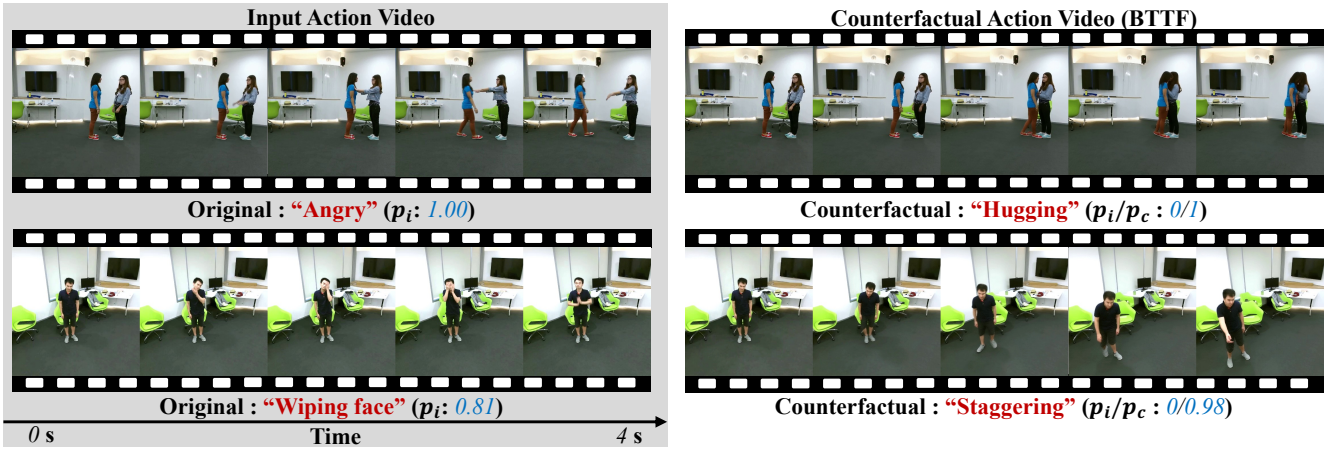

Figure S2. **Video CFEs generated by BTTF for the target video classifier A-swinR trained on NTU RGB+D.**

Table S3. **Quantitative comparison of methods.** BTTF is evaluated against three adapted versions of the image-based classifier-guidance (CG) strategy to validate the limitations of image-based CFE methods to video. CG-Frame applies classifier-guidance independently to each video frame; CG-Video-Mid uses video-classifier gradients but begins denoising from a half-noisy state; and CG-Video performs full-noise initialization with video-level guidance. The benchmark dataset and the target classifier are Shape-Moving and M-swin, respectively.

Method	FR (↑)	SSIM (↑)	LPIPS (↓)	FID (↓)	FVD (↓)
CG-Frame	0.28	0.90	0.28	160.48	2072.14
CG-Video-Mid	1.00	0.95	**0.14**	145.22	1796.21
CG-Video	**1.00**	0.94	0.24	68.01	682.23
BTTF (ours)	0.97	**0.95**	0.20	**27.49**	**250.11**

References

[1] Maximilian Augustin, Valentyn Boreiko, Francesco Croce, and Matthias Hein. Diffusion visual counterfactual explanations. *Advances in Neural Information Processing Systems*, 35:364–377, 2022. 3

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 3

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1

[4] 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. *Advances in neural information processing systems*, 30, 2017. 3

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[6] Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. Diffusion models for counterfactual explanations. In *Proceedings*

of the Asian conference on computer vision, pages 858–876, 2022. 3

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 3

[8] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3202–3211, 2022. 2

[9] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. 2

[10] Bartlomiej Sobieski, Jakub Grzywaczewski, Bartłomiej Sadlej, Matthew Tivnan, and Przemyslaw Biecek. Rethinking visual counterfactual explanations through region constraint. In *The Thirteenth International Conference on Learning Representations*, 2024. 3

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1

[12] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1

[13] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020. 1

[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 3

[15] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 3

[16] Kaisiyuan Wang, Qianyi Wu, Linsen Song, Zhuoqian Yang, Wayne Wu, Chen Qian, Ran He, Yu Qiao, and Chen Change Loy. Mead: A large-scale audio-visual dataset for emotional talking-face generation. In *European conference on computer vision*, pages 700–717. Springer, 2020. 2

[17] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 2

[18] Nina Weng, Paraskevas Pegios, Eike Petersen, Aasa Feragen, and Siavash Bigdeli. Fast diffusion-based counterfactuals for shortcut removal and generation. In *European Conference on Computer Vision*, pages 338–357. Springer, 2024. 3

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3