APPENDIX

In this appendix, we discuss the following details, which could not be included in the main paper owing to space constraints.

A. *ImageCFGen* Algorithm

As described in Section 3.1, our method involves two components: learning the Attribute-SCM ($\mathcal{M}_a$) that models relationships between attributes, and learning the generator ($\mathcal{M}_x$) that produces counterfactual images given the attributes.

A.1. Learning the Attribute-SCM

We assume that the causal graph structure is provided by the user, but the causal functions relating the attributes are not available. For each attribute a_i, the graph entails a set of attributes P_{a_i} that cause the attribute (its *parents* in the graph structure) that should be included while estimating the value of the attribute. It leads to the following generating equation for each attribute,

$$a_i = g_i(P_{a_i}, \epsilon_i) \tag{8}$$

where P_{a_i} are the parents of attribute a_i and ϵ_i is independent noise. The goal is to learn the unknown function g_i for each attribute. Thus, given n attributes, we obtain a set of n equations, each of which can be independently estimated using Maximum Likelihood Estimation from Bayesian networks [8].

A.2. Generating Counterfactuals

Algorithm 1 below shows how *ImageCFGen* is implemented, in extension to the discussion in Sec 3.2 of the main paper. Here we assume that the Attribute-SCM, the Encoder E, and Generator G are pre-learnt and provided as input to the algorithm. $(\mathbf{x}^*, \mathbf{a}^*)$ denotes the input value of the image and the attributes. Note that the Attribute SCM operations (Steps 1-3) are based on the formal procedure to generate counterfactuals from [33].

In Step 1 (Abduction), the encoder uses the input data $(\mathbf{x}^*, \mathbf{a}^*)$ to create the latent vector $\mathbf{z}$.

In Step 2 (Action), given the indices K of sensitive attributes, the goal is to remove the dependence of these sensitive attributes on any other attribute and then change their value to the desired value. To do so, the causal graph of Attribute-SCM is modified to remove all incoming arrows to $\mathbf{a}_K$. That is, the structural equations for each $a_i \in \mathbf{a}_K$ changes from $a_i = g_i(P_{a_i}, \epsilon_i)$ to $a_i = \epsilon_i'$, yielding a modified Attribute-SCM $\mathcal{M}_a'$. Then each $a_i \in \mathbf{a}_K$ is set to the desired modified value a_i^{**} (Line 5; we do this without changing its parents or other variables, hence it is called an *intervention*).

In Step 3 (Prediction), we propagate the change in sensitive attributes to all attributes caused by them. That is,

Algorithm 1: *ImageCFGen*

Input: Input data-attribute pair $(\mathbf{x}^*, \mathbf{a}^*)$; Indices of sensitive attributes to modify K; Modified attribute values $\mathbf{a}_K^{**}$, Attribute SCM $\mathcal{M}_a = \{g_i(P_{a_i^*}, \hat{\epsilon}_i)\} \forall a_i^* \in \mathbf{a}^*, P_{a_i^*} \subseteq \mathbf{a}^*$ where $P_{a_i^*}$ denotes parents of a_i^* and $\hat{\epsilon}_i$ is estimated from data, Encoder E, Generator G

Output: $\mathbf{x}_c$

1 **Step 1: Abduction**
2 $\mathbf{z} = E(\mathbf{x}^*, \mathbf{a}^*)$
3 **Step 2: Action**
4 Remove all incoming arrows to $\mathbf{a}_K$ in $\mathcal{M}_a$ to yield $\mathcal{M}_a'$
5 $a_i \rightarrow a_i^{**} \quad \forall i \in K, a_i^{**} \in \mathbf{a}_K^{**}$ /* Intervene on sensitive attributes and set to modified attribute values */
6 **Step 3: Prediction**
7 $D = desc(\mathbf{a}_K)$ /* descendants of $\mathbf{a}_K$ */
8 $currset = children(\mathbf{a}_K)$
9 $changed[\mathbf{a}^*] = False$
10 **while** *currset is not empty* **do**
11 $cs = \emptyset$;
12 **foreach** $a_j \in currset$ **do**
/* Proceed only if the values of all parents of a_j are already changed */
13 **if** *not* $any_l(P_{a_j}^{(l)} \in D$ & $changed[P_{a_j}^{(l)}] == False)$ **then**
14 $a_j^{**} \rightarrow g_j(P_{a_j^{**}}, \hat{\epsilon_j})$
15 $cs.append(children(a_j))$
16 $changed[a_j] = True$
17 **end**
18 **end**
19 $currset = cs$ /* If there exist children of the current nodes, repeat while loop */
20 **end**
21 $\mathbf{a}_c = \mathbf{a}_K^{**} \cup \mathbf{a}_{-K}^{**}$ /* $\mathbf{a}_{-K}^{**}$ denotes the final values for all non-sensitive attributes */
22 $\mathbf{x}_c = G(\mathbf{z}, \mathbf{a}_c)$

values of all descendants of sensitive attributes $\mathbf{a}_K$ in the causal graph of Attribute-SCM $\mathcal{M}_a'$ are changed, based on the modified value of the sensitive attributes from Step 2. We do so level-by-level: first changing the values of children of $\mathbf{a}_K$ based on the modified value of $\mathbf{a}_K$, then the values of children of children of $\mathbf{a}_K$, and so on. Note that line 13 ensures that an attribute's value is changed only when all its parents' values have been modified. The corresponding equation for each descendant a_j is given in Line 14,

$$a_j^{**} \rightarrow g_j(P_{a_j^{**}}, \hat{\epsilon_j}) \tag{9}$$

where $P_{a_j^{**}}$ refers to the modified values of the parents of a_j, g_j is the pre-learnt Attribute-SCM function, and $\hat{\epsilon_j}$ is

the estimated error term from the original attribute data (i.e., it is the same error value that satisfies $a_j^* \rightarrow g_j(P_{a_j^*}, \hat{\epsilon_j})$). Finally, the updated values of all attributes are used to create $\mathbf{a}_c$, the counterfactual attribute vector. This counterfactual attribute vector is then combined with latent z from Step 1 and provided as input to the generator G, to obtain the counterfactual image $\mathbf{x}_c$ (Line 22).

B. Counterfactual Generation Proof

Proposition 1. *Assume that the true SCM $\mathcal{M}$ belongs to a class of SCMs where the structural equations can be identified uniquely given the causal graph. Further, assume that the Encoder E, Generator G and Attribute SCM are optimal and thus correspond to the true structural equations of the SCM $\mathcal{M}$. Then Equation 1 generates a valid counterfactual image for any given input $(\mathbf{x}, \mathbf{a})$ and the requested modified attributes $\mathbf{a}'_k$.*

Proof. Let $\mathcal{M} = \{\mathcal{M}_x, \mathcal{M}_a\}$ be the true SCM that generates the data $(\mathbf{x}, \mathbf{a})$. Let $\mathbf{a}_k \subset \mathbf{a}$ be the attributes you want to intervene on. Let $\mathbf{a}_{-k} = \mathbf{a} \setminus \mathbf{a}_k$ be the remaining attributes. The corresponding equations for $\mathcal{M}$ are:

$$\begin{aligned} a_i &:= g_i(p_{a_i}, \epsilon_i), \quad \forall i = 1..n \\ \mathbf{x} &:= g(\mathbf{a}_k, \mathbf{a}_{-k}, \epsilon) \end{aligned} \tag{10}$$

where ϵ and ϵ_i refer to independent noise, g and g_i refer to structural assignments of the SCMs $\mathcal{M}_x$ and $\mathcal{M}_a$ respectively while p_{a_i} refers to parent attributes of a_i. Given an input $(\mathbf{x}, a_k, a_{-k})$, we generate a counterfactual using $\mathcal{M}$ and show that it is equivalent to Equation 1.

Abduction: Infer ϵ for $\mathbf{x}$ and ϵ_i for all a_i from Equation 10 using the following equations:

$$\begin{aligned} \hat{\epsilon_i} &:= g_i^{-1}(a_i, p_{a_i}), \quad \forall i = 1..n \\ \hat{\epsilon} &:= g^{-1}(\mathbf{x}, \mathbf{a}_k, \mathbf{a}_{-k}) \end{aligned} \tag{11}$$

Action: Intervene on all attributes $\mathbf{a}_k$ by setting them to the requested modified attributes $\mathbf{a}'_k$.

$$a_i \rightarrow a'_i \quad \forall a_i \in \mathbf{a}_k \quad \text{and} \quad \forall a'_i \in \mathbf{a}'_k \tag{12}$$

Prediction: The following equation then changes the values of all descendants of $\mathbf{a}_k$.

$$desc(a_i) \rightarrow g_i(p_{desc(a_i)}, \hat{\epsilon_i}) \quad \forall a_i \in \mathbf{a}_k \tag{13}$$

where $desc(a_i)$ are descendants of a_i and $p_{desc(a_i)}$ are parents of the descendants of a_i, $\forall a_i \in \mathbf{a}_k$. Let $\mathbf{a}_c = \mathbf{a}'_k \cup \mathbf{a}'_{-k}$ where $\mathbf{a}'_{-k}$ are (possibly) modified values of the other attributes according to Equation 13. Therefore, the counterfactual of $\mathbf{x}$, $\mathbf{x}_c$ can be generated as:

$$\mathbf{x}_c := g(\mathbf{a}_c, \hat{\epsilon}) \tag{14}$$

We now show that Equation 1 produces the same $\mathbf{x}_c$. By the assumption in the theorem statement, the Attribute-SCM corresponds to the structural assignments $\{g_i, g_i^{-1}\}$, $\forall i = 1..n$ of SCM $\mathcal{M}_a$ while the Generator G learns the structural assignment g and the Encoder E learns g^{-1} of the SCM $\mathcal{M}_x$. Hence, the Attribute-SCM, Generator and Encoder learn the combined SCM $\mathcal{M}$.

When the SCM assignments learned by the Attribute-SCM are optimal, i.e. Attribute-SCM = $\mathcal{M}_a$ then:

$$\hat{a}_c = a_c$$

Similarly, under optimal Generator, $G = g$ and $E = g^{-1}$:

$$\begin{aligned} \mathbf{x}_c &= g(\mathbf{a}_c, g^{-1}(\mathbf{x}, \mathbf{a}_k, \mathbf{a}_{-k})) \\ &= G(\mathbf{a}_c, E(\mathbf{x}, \mathbf{a})) \quad (\text{as} \quad \mathbf{a} = \mathbf{a}_k \cup \mathbf{a}_{-k}) \end{aligned} \tag{15}$$

which is the same as Equation 1. □

C. Counterfactual Fairness Proof

Definition 1. *Counterfactual Fairness from [24]. Let $\mathcal{A}$ be the set of attributes, comprising of sensitive attributes $\mathcal{A}_S \subseteq \mathcal{A}$ and other non-sensitive attributes $\mathcal{A}_N$. The classifier $\hat{f}$ is counterfactually fair if under any context $\mathbf{X} = \mathbf{x}$ and $\mathbf{A} = \mathbf{a}$, changing the value of the sensitive features to $\mathbf{A}_S \leftarrow a'_s$ counterfactually does not change the classifier's output distribution Y.*

$$\begin{aligned} &P(Y_{A_S \leftarrow a_s} = y | \mathbf{X} = \mathbf{x}, \mathbf{A}_S = \mathbf{a}_s, \mathbf{A}_N = \mathbf{a}_N) \\ &= P(Y_{A_S \leftarrow a'_s} = y | \mathbf{X} = \mathbf{x}, \mathbf{A}_S = \mathbf{a}_s, \mathbf{A}_N = \mathbf{a}_N) \end{aligned} \tag{16}$$

for all y and for any value $\mathbf{a}'_s$ attainable by $\mathbf{A}_S$.

Proposition 2. *Under the assumptions of Proposition 1 for the encoder E, generator G, and Attribute SCM $\mathcal{M}_a$, a classifier $\hat{f}(\mathbf{X}) : \mathcal{X} \rightarrow \mathcal{Y}$ that satisfies zero bias according to Equation 4 is counterfactually fair with respect to $\mathcal{M}$.*

Proof. To evaluate fairness, we need to reason about the output $Y = y$ of the classifier $\hat{f}$. Therefore, we add a functional equation to the SCM $\mathcal{M}$ from Equation 10.

$$\begin{aligned} a_i &:= g_i(p_{a_i}, \epsilon_i), \quad \forall i = 1..n \\ \mathbf{x} &:= g(\mathbf{a}_k, \mathbf{a}_{-k}, \epsilon) \\ y &\leftarrow \hat{f}(\mathbf{x}) \end{aligned} \tag{17}$$

where ϵ and ϵ_i are independent errors as defined in Equation 10, and $P_{\mathbf{a}_i}$ refers to the parent attributes of an attribute a_i as per the Attribute-SCM. The SCM equation for y does not include an noise term since the ML classifier $\hat{f}$ is a deterministic function of $\mathbf{X}$. In the above equations, the attributes $\mathbf{a}$ are separated into two subsets: $\mathbf{a}_k$ are the attributes specified to be changed in a counterfactual and $\mathbf{a}_{-k}$ refers to all other attributes.

Based on this SCM, we now generate a counterfactual $y_{a_k \leftarrow a'_k}$ for an arbitrary new value a'_k. Using the **Prediction** step, the counterfactual output label y_c for an input $(\mathbf{x}, \mathbf{a})$ is given by: $y_c = \hat{f}(\mathbf{X} = \mathbf{x}_c)$. From Theorem 1, under optimality of the encoder E, generator G and learned functions g_i, we know that $\mathbf{x}_c$ generated by the following equation is a valid counterfactual for an input $(\mathbf{x}, \mathbf{a})$,

$$\begin{aligned} \mathbf{x}_c &= G(E(\mathbf{x}, \mathbf{a}), \mathbf{a}_c) \\ &= X_{\mathbf{A}_k \leftarrow a'_k} | (\mathbf{X} = \mathbf{x}, \mathbf{A}_k = \mathbf{a}_k, \mathbf{A}_{-k} = \mathbf{a}_{-k}) \end{aligned} \tag{18}$$

where $\mathbf{a}_c$ represents the modified values of the attributes under the action $\mathbf{A}_k \leftarrow a'_k$. Therefore, $y_{a_k \leftarrow a'_k} | (\mathbf{X} = \mathbf{x}, \mathbf{A}_k = \mathbf{a}_k, \mathbf{A}_{-k} = \mathbf{a}_{-k})$ is given by $y_c = \hat{f}(\mathbf{x}_c)$.

Using the above result, we now show that the bias term from Equation 4 and 5 reduces to the counterfactual fairness definition from Definition 1,

$$\begin{aligned} &P(y_r = 0, y_c = 1) - P(y_r = 1, y_c = 0) \\ &\quad = [P(y_r = 0, y_c = 1) + P(y_r = 1, y_c = 1)] - \\ &\qquad [P(y_r = 1, y_c = 0) + (P(y_r = 1, y_c = 1)] \\ &\quad = P(y_c = 1) - P(y_r = 1) \\ &\quad = [P(Y_{A_k \leftarrow a'_k} = 1 | \mathbf{X} = \mathbf{x}, \mathbf{A}_k = \mathbf{a}_k, \mathbf{A}_{-k} = \mathbf{a}_{-k}) - \\ &\qquad P(Y_{A_k \leftarrow a_k} = 1 | \mathbf{X} = \mathbf{x}, \mathbf{A}_k = \mathbf{a}_k, \mathbf{A}_{-k} = \mathbf{a}_{-k})] \\ &\quad = [P(Y_{A_S \leftarrow a'_s} = 1 | \mathbf{X} = \mathbf{x}, \mathbf{A}_S = \mathbf{a}_s, \mathbf{A}_N = \mathbf{a}_n) - \\ &\qquad P(Y_{A_S \leftarrow a_s} = 1 | \mathbf{X} = \mathbf{x}, \mathbf{A}_S = \mathbf{a}_s, \mathbf{A}_N = \mathbf{a}_n)] \end{aligned} \tag{19}$$

where the second equality is since $y_r, y_c \in \{0, 1\}$, the third equality is since the reconstructed y_r is the output prediction when $\mathbf{A} = \mathbf{a}$, and the last equality is $\mathbf{A}_k$ being replaced by the sensitive attributes $\mathbf{A}_S$ and $\mathbf{A}_{-k}$ being replaced by $\mathbf{A}_N$. We can prove a similar result for $y_c = 0$. Hence, when bias term is zero, the ML model $\hat{f}$ satisfies counterfactual fairness (Definition 1). □

D. Architecture Details

Here we provide the architecture details for the base Adversarially Learned Inference (ALI) model trained on Morpho-MNIST and CelebA datasets. For Cyclic Style ALI, we replace the ALI generator with the style-based generator architecture in [20]. We however do not use progressive growing or other regularization strategies suggested in [20, 21] while training our model. For details on the style-based generation architecture, please refer [20].

Overall, the architectures and hyperparameters are similar to the ones used by [11], with minor variations. Instead of using the Embedding network from the original paper, the attributes are directly passed on to the Encoder, Generator and Discriminator. We found that this enabled better conditional generation in our experiments. All experiments were implemented using Keras 2.3.0 [5] with Tensorflow 1.14.0 [1]. All models were trained using a Nvidia Tesla P100 GPU.

D.1. Morpho-MNIST

Tables 4, 5 and 6 show the Generator, Encoder and Discriminator architectures respectively for generating Morpho-MNIST counterfactuals. Conv2D refers to Convolution 2D layers, Conv2DT refers to Transpose Convoloution layers, F refers to number of filters, K refers to kernel width and height, S refers to strides, BN refers to Batch Normalization, D refers to dropout probability and A refers to the activation function. LReLU denotes the Leaky ReLU activation function. We use the Adam optimizer [22] with a learning rate of 10^{-4}, $\beta_1 = 0.5$ and a batch size of 100 for training the model. For the LeakyReLU activations, $\alpha = 0.1$. The model converges in approximately 30k iterations. All weights are initialized using the Keras truncated normal initializer with $mean = 0.0$ and $stddev = 0.01$. All biases are initialized with zeros.

Layer	F	K	S	BN	D	A
Conv2DT	256	(4,4)	(1,1)	Y	0.0	LReLU
Conv2DT	128	(4,4)	(2,2)	Y	0.0	LReLU
Conv2DT	64	(4,4)	(1,1)	Y	0.0	LReLU
Conv2DT	32	(4,4)	(2,2)	Y	0.0	LReLU
Conv2DT	32	(1,1)	(1,1)	Y	0.0	LReLU
Conv2D	1	(1,1)	(1,1)	Y	0.0	Sigmoid

Table 4: **Architecture for Morpho-MNIST Generator**

Layer	F	K	S	BN	D	A
Conv2D	32	(5,5)	(1,1)	Y	0.0	LReLU
Conv2D	64	(4,4)	(2,2)	Y	0.0	LReLU
Conv2D	128	(4,4)	(1,1)	Y	0.0	LReLU
Conv2D	256	(4,4)	(2,2)	Y	0.0	LReLU
Conv2D	512	(3,3)	(1,1)	Y	0.0	LReLU
Conv2D	512	(1,1)	(1,1)	Y	0.0	Linear

Table 5: **Architecture for Morpho-MNIST Encoder**

Layer	F	K	S	BN	D	A
D_z						
Conv2D	512	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	512	(1,1)	(1,1)	N	0.5	LReLU
D_x						
Conv2D	32	(5,5)	(1,1)	N	0.2	LReLU
Conv2D	64	(4,4)	(2,2)	Y	0.2	LReLU
Conv2D	128	(4,4)	(1,1)	Y	0.5	LReLU
Conv2D	256	(4,4)	(2,2)	Y	0.5	LReLU
Conv2D	512	(3,3)	(1,1)	Y	0.5	LReLU
$D_{x\text{-}z}$						
Conv2D	1024	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	1024	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	1	(1,1)	(1,1)	N	0.5	Sigmoid

Table 6: **Architecture for Morpho-MNIST Discriminator.** D_x, D_z and D_{xz} are the discriminator components to process the image x, the latent variable z and the output of D_x and D_z concatenated, respectively.

D.2. CelebA

Tables 7, 8 and 9 show the Generator, Encoder and Discriminator architectures respectively for generating CelebA counterfactuals.

Layer	F	K	S	BN	D	A
Conv2DT	512	(4,4)	(1,1)	Y	0.0	LReLU
Conv2DT	256	(7,7)	(2,2)	Y	0.0	LReLU
Conv2DT	256	(5,5)	(2,2)	Y	0.0	LReLU
Conv2DT	128	(7,7)	(2,2)	Y	0.0	LReLU
Conv2DT	64	(2,2)	(1,1)	Y	0.0	LReLU
Conv2D	3	(1,1)	(1,1)	Y	0.0	Sigmoid

Table 7: **Architecture for CelebA Generator**

Layer	F	K	S	BN	D	A
Conv2D	64	(2,2)	(1,1)	Y	0.0	LReLU
Conv2D	128	(7,7)	(2,2)	Y	0.0	LReLU
Conv2D	256	(5,5)	(2,2)	Y	0.0	LReLU
Conv2D	256	(7,7)	(2,2)	Y	0.0	LReLU
Conv2D	512	(4,4)	(1,1)	Y	0.0	LReLU
Conv2D	512	(1,1)	(1,1)	Y	0.0	Linear

Table 8: **Architecture for CelebA Encoder**

Layer	F	K	S	BN	D	A
D_z						
Conv2D	1024	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	1024	(1,1)	(1,1)	N	0.2	LReLU
D_x						
Conv2D	64	(2,2)	(1,1)	N	0.2	LReLU
Conv2D	128	(7,7)	(2,2)	Y	0.2	LReLU
Conv2D	256	(5,5)	(2,2)	Y	0.2	LReLU
Conv2D	256	(7,7)	(2,2)	Y	0.2	LReLU
Conv2D	512	(4,4)	(1,1)	Y	0.2	LReLU
D_{x_z}						
Conv2D	2048	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	2048	(1,1)	(1,1)	N	0.2	LReLU
Conv2D	1	(1,1)	(1,1)	N	0.2	Sigmoid

Table 9: **Architecture for CelebA Discriminator.** D_x, D_z and D_{xz} are the discriminator components to process the image x, the latent variable z and the output of D_x and D_z concatenated, respectively.

E. Morpho-MNIST Latent Space Interpolations

We also plot latent space interpolations between pairs of images sampled from the test set. Figure 8 shows that the model has learned meaningful latent space representations and the transitional images look realistic as well.

F. Morpho-MNIST Reconstructions

We qualitatively evaluate the inference model (Encoder) by sampling images along with their attributes from the test set and passing them through the encoder to obtain their latent space representations. These representations are passed to the generator which outputs reconstructions of the original image. The reconstructions are showed in Figure 9. Overall, reconstructions for Morpho-MNIST are faithful reproductions of the real images.

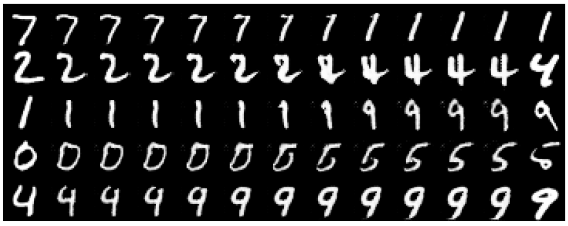

Figure 8: **Morpho-MNIST Interpolations.** The columns on the extreme left and right denote real samples from the test set and the columns in between denote generated images for the linearly interpolated latent space representation **z**.

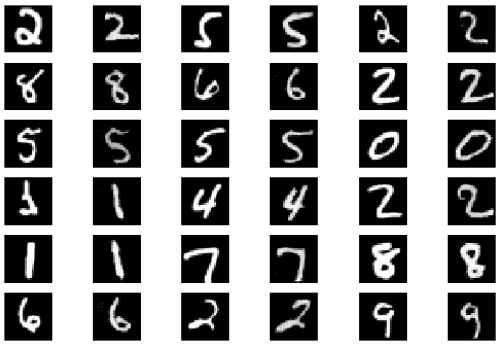

Figure 9: **Morpho-MNIST Reconstructions.** Odd columns denote real images sampled from the test set, and even columns denote reconstructions for those real images.

G. Morpho-MNIST: Evaluating Label CFs

As shown in Figure 3, we empirically evaluated the counterfactual generation on Morpho-MNIST by generating CFs that change the digit label for an image. To check whether the generated counterfactual image corresponds to the desired digit label, we use the output of a digit classifier trained on Morpho-MNIST images. Here we provide details of this pre-trained digit classifier. The classifier architecture is shown in Table 10. The classifier converges in approximately 1k iterations with a validation accuracy of 98.30%. We then use the classifier to predict the labels of the counterfactual images and compare them to the labels that they were supposed to be changed to (target label). Overall, 97.30% of the predicted labels match the target label. Since the classifier is not perfect, the difference be-

Layer	F	K	S	BN	D	A
Conv2D	32	(3,3)	(1,1)	N	0.0	ReLU
Conv2D	64	(3,3)	(1,1)	N	0.0	ReLU
MaxPool2D	-	(2,2)	-	N	0.0	-
Dense	256	-	-	N	0.5	ReLU
Dense	128	-	-	N	0.5	ReLU
Dense	10	-	-	N	0.5	Softmax

Table 10: **Morpho-MNIST Label Classifier**

tween the CF image's (predicted) label and the target label may be due to an error in the classifier's prediction or due to an error in CF generation, but this is minimal. We show some images for which predicted labels do not match the target labels in Figure 10. Most of these images are digits that can be confused as another digit; for instance, the first image in row 2 is a counterfactual image with a target label of 9, but was predicted by the digit classifier as a 1.

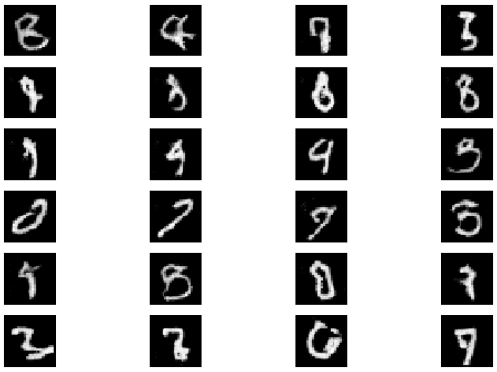

Figure 10: **Misclassified Counterfactuals.** Counterfactual images for Morpho-MNIST on which target label does not match predicted label.

H. DeepSCM Implementation on CelebA

Since DeepSCM was not implemented on CelebA, we reproduce their method based on their available code and choose an appropriate VAE architecture for CelebA data. Tables 11 and 12 show the Decoder and Encoder architectures respectively for generating CelebA counterfactuals. Specifically, we use Equation A.4 from the DeepSCM paper [32] to model the image $\mathbf{x}$. We use a conditional VAE with a fixed variance Gaussian decoder to output a *bias* to further reparametrize a Gaussian distribution using a location-scale transform (as stated in [32]). In our implementation of DeepSCM, we do not need to model the attributes $\mathbf{a}$ using Normalizing Flows, since the attributes independently influence the images in the underlying SCM in case of CelebA (as explained in Section 5). We set the latent dimension to 256, use the Adam optimizer [22] with a learning rate of 0.0005 to train the conditional VAE with a batch size of 128. Similar to Pawloski *et al*. [32], we adopt a constant variance assumption and set $\log \sigma^2 = -5$. We also preprocess the images by scaling them between [0,1); the preprocessing flow in Equation A.4 in [32] is hence altered by normalizing its outputs between [0,1).

Layer	F	K	S	BN	D	A
Dense	4096	-	-	N	0.0	Linear
Conv2DT	256	(3,3)	(2,2)	Y	0.25	LReLU
Conv2DT	128	(3,3)	(2,2)	Y	0.25	LReLU
Conv2DT	64	(3,3)	(2,2)	Y	0.25	LReLU
Conv2DT	32	(3,3)	(2,2)	Y	0.25	LReLU
Conv2D	3	(3,3)	(1,1)	Y	0.0	Sigmoid

Table 11: **Architecture for CelebA DeepSCM Decoder**

Layer	F	K	S	BN	D	A
Conv2D	32	(3,3)	(2,2)	Y	0.25	LReLU
Conv2D	64	(3,3)	(2,2)	Y	0.25	LReLU
Conv2D	128	(3,3)	(2,2)	Y	0.25	LReLU
Conv2D	256	(3,3)	(2,2)	Y	0.25	LReLU
Conv2D	256	(1,1)	(1,1)	N	0.0	LReLU

Table 12: **Architecture for CelebA DeepSCM Encoder**

I. Additional DeepSCM vs *ImageCFGen* CelebA Counterfactuals

In addition to Figure 5 of the main paper, we provide additional counterfactual images obtained using *ImageCFGen* and DeepSCM in Figure 11. As observed in Section 5.2, the counterfactuals generated using *ImageCFGen* are more true to the corresponding intervention than DeepSCM.

J. Additional *ImageCFGen* CelebA Counterfactuals

Figure 12 shows more CF examples obtained using *ImageCFGen*. Note how the CF image is different w.r.t. the base (reconstructed) image only in terms of the intervened attribute. Consequently, if the attribute in the base image is already present, the CF image is exactly the same as the original image. For instance, (Ib) and (Ig) in Fig 12 are exactly the same since (Ia) already has brown hair, hence intervening on brown hair has no effect on the hair color.

K. Ablation Study of Style-Based Generator and Cyclic Cost Minimization

We plot the reconstructed images of real images, produced by ALI, StyleALI (ALI with style based generator) and Cyclic Style ALI (Style ALI with Cyclic Cost Minimization) in Figure 13. We observe that using a style-based generator significantly improves the quality of the generated

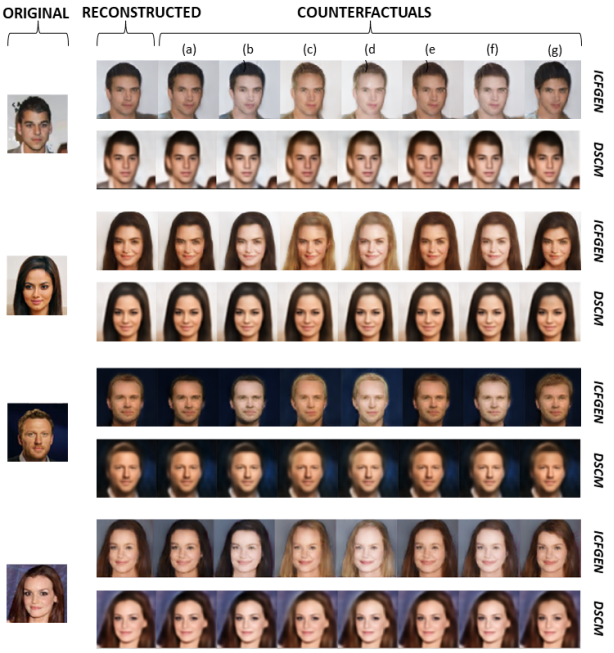

Figure 11: ***ImageCFGen* and DeepSCM Counterfactuals.** (a) denotes do (black hair = 1) and (b) denotes do (black hair = 1, pale =1). Similarly (c) denotes do (blond hair = 1); (d) denotes do (blond hair = 1, pale = 1); (e) denotes do (brown hair = 1); (hf denotes do (brown hair = 1, pale = 1); and (g) denotes do (bangs = 1).

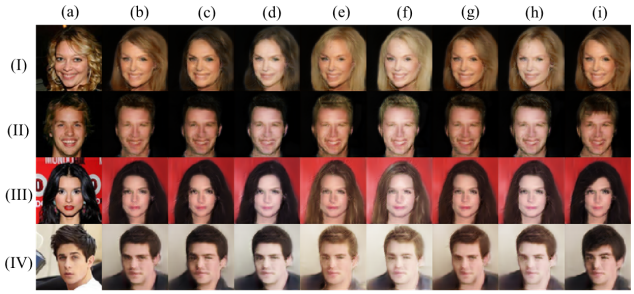

Figure 12: **CelebA Counterfactuals.** Column (a) across all rows I, II, III, IV represent the real image and (b) represents the reconstructed image. (c) denotes do (black hair = 1) and (d) denotes do (black hair = 1, pale =1). Similarly (e) denotes do (blond hair = 1); (f) denotes do (blond hair = 1, pale = 1); (g) denotes do (brown hair = 1); (h) denotes do (brown hair = 1, pale = 1); and (i) denotes do (bangs = 1).

images and applying the cyclic cost minimization algorithm on top of it improves reconstruction of the real image.

L. Human Evaluation of Counterfactuals

To quantitatively evaluate the counterfactuals, we asked human evaluators to pick the "edited" (counterfactual) image from 10 randomly sampled pairs of reconstructed and

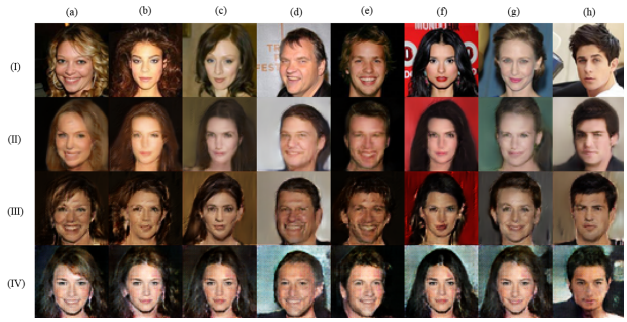

Figure 13: **Ablation Study.** Row (I) consists of real images, (II) consists of reconstructions generated by Cyclic Style ALI (CSALI), (III) Style ALI reconstructions and (IV) has ALI reconstructions

counterfactual images to human evaluators and asked them to pick the *edited* (counterfactual) image. Overall, we got 66 responses resulting in an average score of 5.15 correct answers out of 10 with a standard deviation of 1.64, indicating that the generated counterfactual distribution is perceptually indistinguishable from the reconstructed one. Figure 14 shows sample questions on the form circulated for our human evaluation studies. For each question, the user chose one of two images that seems edited to the human eye. If the counterfactual can fool human perception, it indicates better performance of counterfactual generation.'

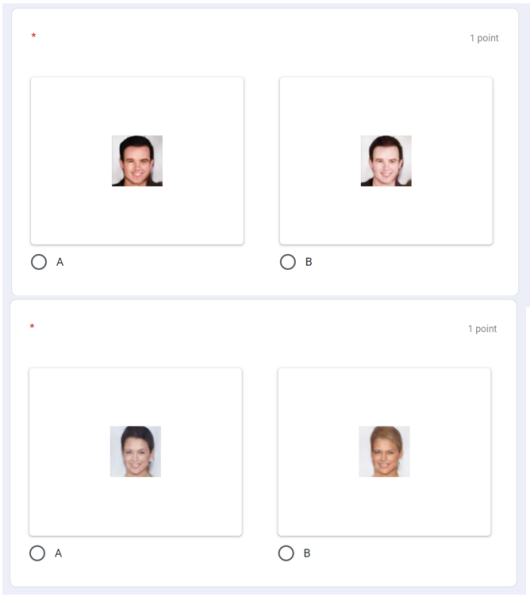

Figure 14: **Edited or Not?** Sample questions on the form circulated for human evaluation studies

M. *Attractive* Classifier Architecture

We describe the architecture and training details for the *attractiveness* classifier whose fairness was evaluated in Figure 6. The same classifier's output was explained in Figure 7. *Attractive* is a binary attribute associated with every image in CelebA dataset. The architecture for the binary classifier is shown in Table 13. We use the Adam optimizer

Layer	F	K	S	BN	D	A
Conv2D	64	(2,2)	(1,1)	Y	0.2	LReLU
Conv2D	128	(7,7)	(2,2)	Y	0.2	LReLU
Conv2D	256	(5,5)	(2,2)	Y	0.2	LReLU
Conv2D	256	(7,7)	(2,2)	Y	0.2	LReLU
Conv2D	512	(4,4)	(1,1)	Y	0.5	LReLU
Conv2D	512	(1,1)	(1,1)	Y	0.5	Linear
Dense	128	-	-	N	0.5	ReLU
Dense	128	-	-	N	0.5	ReLU
Dense	1	-	-	N	0.5	Sigmoid

Table 13: **Architecture for the *Attractiveness* Classifier**

with a learning rate of 10^{-4}, $\beta_1 = 0.5$ and a batch size of 128 for training the model. For the LeakyReLU activations, $\alpha = 0.02$. The model converges in approximately 20k iterations. All weights are initialized using the Keras truncated normal initializer with $mean = 0.0$ and $stddev = 0.01$. All biases are initialized with zeros.

N. Bias Mitigation Details

We now present more details of our bias mitigation results in continuation to the discussion in Sec 5.4 of the main paper. To pick the optimal model while finetuning the classifier with the proposed bias mitigation regularizer, we set an accuracy threshold of 80% and pick the model with the lowest bias. We use $\lambda = 1.0$ and use an Adam optimizer with a learning rate of 10^{-4}, $\beta_1 = 0.5$ and a batch size of 128 for training the model.

	Fair Classifier	Biased Classifier
black_h, pale	0.032	0.159
blond_h, pale	-0.041	0.077
brown_h, pale	0.012	0.154

Table 14: **Bias Values after Bias Mitigation.** Lower bias is better. Absolute bias values less than 5% are not considered significant.

O. Complex Attribute SCM

Our method can also be utilized in the setting where the attributes are connected in a complex causal graph structure, unlike [7, 17]. We now conduct a fairness analysis w.r.t age for the *Attractive* classifier, assuming that *Young* affects other visible attributes like *Gray hair*.

Defining the Causal Graph. Fig 15 shows a potential causal graph for the attribute *Young*. While we ignored the attribute *Young* in the previous experiments, we now propose a SCM that describes the relationship between *Young* and a given image, mediated by a few facial attributes. Among the two demographic attributes, we choose *Young* over *Male* since *Young* has some descendants that can be agreed upon in general, whereas using *Male* would have forced us to explicitly model varying societal conventions surrounding this attribute. For e.g., the SCM proposed by [23] describes a causal relationship from *gender* to *smiling* and *narrow eyes*, which is a problematic construct. For the attribute *Young*, we specifically choose *Gray Hair*, *Eyeglasses* and *Receding Hairline* since it is reasonable to assume that older people are more likely to have these attributes, as compared to younger ones.

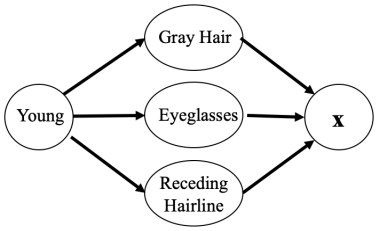

Figure 15: ***Young* Attribute-SCM.** SCM that describes the relationship between the attribute *Young* and the generated image.

Learning the Attribute SCM. To learn the model parameters for the SCM shown in Figure 15, we estimate conditional probabilities for each edge between *Young* and the other attributes using maximum likelihood over the training data, as done in Bayesian networks containing only binary-valued nodes [34]. This SCM in Figure 15, combined with the SCM from Fig 2b that connects other facial attributes to the image, provides us the *augmented* Attribute SCM that we use for the downstream task of CF generation. Once the Attribute SCM is learned, the rest of the steps in Algorithm 1 remain the same as before. That is, based on this augmented Attribute SCM, the counterfactuals are generated according to the Eqn 1. For example, to generate a CF changing *Young* from 1 to 0, the Prediction step involves changing the values of gray hair, receding hairline and eyeglasses based on the modified value of *Young*, according to the learned parameters (conditional probabilities) of the SCM.

Fairness Analysis. We conduct a fairness analysis similar as above, using the above attractiveness classifier. We generate counterfactuals for *Young = 1* and *Young = 0* according to the causal graph in Fig 15. The analysis is showed in Fig 16; we observe that the classifier is evidently biased against *Young = 0* and biased towards *Young = 1*, when predicting *attractive=1*. We quantify this bias using Eqn 5. Using counterfactuals that change *Young* from 1 to 0, we get a negative bias of -0.136 and for changing *Young* from 0 to 1,

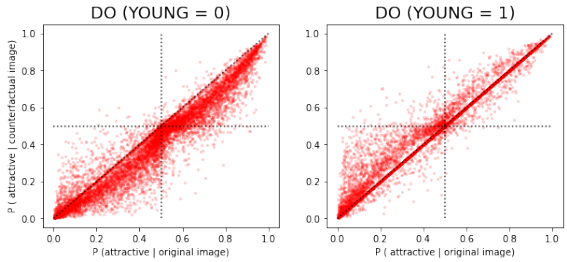

Figure 16: **Fairness Analysis for Complex SCM.** Counterfactual images that change *Young* from 1 to 0 (left), have a lower prediction score from the *attractiveness* classsifier, while counterfactual images that change *Young* from 0 to 1 (right), have a slightly higher prediction score.

we get a positive bias of 0.087 which are both substantial biases assuming a threshold of 5%. Therefore, given the causal graph from Figure 15, our method is able to generate counterfactuals for complex high-level features such as age, and use them to detect any potential biases in a machine learning classifier.

we acknowledge that there are many challenges in generating meaningful and realistic counterfactuals