A Related Work

This section succinctly presents an overview of fairness in the context of machine learning, especially focusing on the burgeoning concept of counterfactual fairness.

Fairness in Machine Learning. While there have been several efforts on developing fair machine learning algorithms (see, e.g., [23, 44, 42, 22, 3, 34]), fairness still remains an elusive and loosely defined concept. Several definitions have been proposed, each motivated by unique considerations and emphasizing different elements. For instance, one such notion is 'unawareness', which necessitates the exclusion of sensitive attributes from the input data fed into machine learning models. Conversely, parity-based fairness sets guidelines on how models should perform across different demographics. Demographic parity, a widely accepted group fairness criterion, ensures consistent distribution of predictions ($\hat{Y}$), regardless of sensitive attributes (A), defined as $P(\hat{Y}|A=0) = P(\hat{Y}|A=1)$ [13].

Equal opportunity is another key criterion, ensuring that individuals from diverse protected attribute groups have the same probability of selection, represented by $P(\hat{Y}=1|A=0,Y=1) = P(\hat{Y}=1|A=1,Y=1)$ [20]. The related concept of equal odds prescribes equal true positive and false positive rates across different protected attribute groups [38].

Additionally, preference-based fairness argues that an algorithm's design should not be solely determined by its creators or regulators but should also incorporate the preferences of those directly affected by the algorithm's outputs [43, 12].

Counterfactual Fairness. Emerging from the broader concept of individual fairness, counterfactual fairness seeks to guarantee that similar datapoints are treated identically [32]. It uses counterfactual pairs to establish similarity. The field of natural language processing has seen the use of intuitive causal models, where words associated with protected attributes are substituted to generate counterfactual data [29, 14]. However, this approach might overlook possible causal relationships among words.

To overcome this, a more sophisticated causal model was proposed by [31], facilitating a three-step process for counterfactual data generation: Abduction-Action-Prediction. This process includes the inference of exogenous variables' distribution based on observed data, the modification of protected attributes, and the computation of resultant attributes. Leveraging this causal model, [26] proposed the formal definition of counterfactual fairness.

Adopting the advanced assumptions of [31] about structural functions in the causal model allows for counterfactual data generation via the Markov chain Monte Carlo (MCMC) algorithm [15]. Several practical applications employ an encoder-decoder-like structure for counterfactual inference, with the encoder's hidden representation considered as the exogenous variable [30, 21, 37, 24]. The decoder predicts counterfactual data after the sensitive attribute is modified. Certain research, such as [8], explores counterfactual inference without a causal model by reconceptualizing it as a multi-objective issue.

A myriad of techniques exist to construct fair models using counterfactual inference. The simplest among these, unawareness, involves the removal of the protected attribute from the input [18]. Yet, due to potential correlations between remaining features and protected attributes, this method often falls short. [26] suggested an approach that uses only the non-descendant variables of the sensitive attribute as model inputs, achieving perfect fairness. Other research aimed to attain approximate counterfactual fairness [36].

A common approach employed by [14, 36, 6, 11, 33] introduces a fairness penalty regularizer to the loss function when training a counterfactually fair model. Meanwhile, studies by [24, 14, 35, 10] have used a counterfactual augmentation method to increase fairness. This involves generating a new training dataset by mixing counterfactual and factual data. Notably, several studies [17, 39, 41, 19, 28] sought to minimize the correlation between exogenous variables and the sensitive attribute using adversarial learning or regularization. They propose another kind of fairness from the causal perspective, differing from [26].

While the method proposed by [26] maintains perfect fairness, it often leads to a significant decrease in precision due to the underutilization of observed data's information. Subsequent methodologies have managed to enhance precision, though they cannot theoretically guarantee counterfactual fairness. As

counterfactual fairness gains increasing traction [17, 6, 27], there is an urgent need for approaches that simultaneously augment performance and uphold fairness.

B Proofs

Theorem 1. We start by finding $\Pr\{R_{A\leftarrow a'}(U) = r|X = x, A = a\}$.

$$\Pr(R_{A\leftarrow a'}(U) = r|X = x, A = a) = \int \Pr(R_{A\leftarrow a'}(u) = r|X = x, A = a) \Pr(U = u|X = x, A = a)$$

Note that R in Algorithm 1 depends on value realization u and feature vector x. However, given u, $R_{A\leftarrow a'}(u)$ does not depend on intervention $A = a'$ as $s(.)$ is a symmetric function and gets all the counterfactual samples for different values of a'. As a result, $\Pr(R_{A\leftarrow a'}(U) = r|X = x, A = a)$ does not depend on a'. Moreover, $\Pr(U = u|X = x, A = a)$ is not a function of a' and does not depend on a'. This implies the right-hand side of (3) does not change by a'. As a result, for a given pair of (x, a), $\Pr(R_{A\leftarrow a'}(U) = r|X = x, A = a)$ remains unchanged for all $a' \in \mathcal{A}$. This implies that R satisfies CF. Consequently, any function of R including $g_w(R)$ satisfies CF. □

Theorem 2. Assume that R has been generated using Algorithm 2. We have,

$$\Pr(R_{A\leftarrow a', X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(U) = r|X = x, A = a) =$$
$$\int \Pr(R_{A\leftarrow a', X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(u) = r|X = x, A = a) \Pr(U = u|X = x, A = a)$$

Note that, $\Pr(R_{A\leftarrow a', X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(U) = r|X = x, A = a)$ does not depend on a'. This is because, Algorithm 2 generates a presentation using a symmetric function, and all the counterfactual samples $\{(\check{x}^{[1]}_{\mathcal{P}_{\mathcal{G}_A}}, \check{a}^{[1]}), \ldots, (\check{x}^{[|\mathcal{A}|-1]}_{\mathcal{P}_{\mathcal{G}_A}}, \check{a}^{[|\mathcal{A}|-1]})\}$, and intervention on A does not change the presentation. Note that $\Pr(U = u|X = x, A = a)$ is not a function of a' and does not depend on a'. This implies the right-hand side of (3) does not change by a'. As a result, for a given pair of (x, a), $\Pr(R_{A\leftarrow a', X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(U) = r|X = x, A = a)$ remains unchanged for all $a' \in \mathcal{A}$. This implies that R satisfies PCF. Consequently, any function of R including $g_w(R)$ satisfies PCF.

$$\Pr\{\hat{Y}_{A\leftarrow a, X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(U) = y|X = x, A = a\} = \Pr\{\hat{Y}_{A\leftarrow a', X_{\mathcal{P}^c_{\mathcal{G}_A}} \leftarrow x_{\mathcal{P}^c_{\mathcal{G}_A}}}(U) = y|X = x, A = a\}$$

□

C Synthetic Data Simulation

For the real data experiments in Section 5, the causal model behind the problem remains unknown, and we had to make an assumption about the causal structure and estimate the causal model parameters using observed data.

In order to make sure that we are working with a true causal model and true structural equations and demonstrate our proposed method can improve performance while maintaining counterfactual fairness, we carry out a simulation experiment on the synthetic data.

We consider a causal graph shown in Figure 6. The structural function defined in the corresponding causal model is

$$f_X = \sin U_1 + \cos U_2 A + A + 0.1; f_Y = 0.2X^2 + 1.2X + 0.2$$

To generate the synthetic dataset, we sampled A from the Bernoulli distribution with $p = 0.4$ for 3000 times. U_1 and U_2 are sampled independently from the normal distribution $\mathcal{N}(0, 1)$. X and Y were computed with the structural function. The counterfactual data $\check{X}$ were computed by substituting A in the structural function with $\check{A}$.

We implemented our method and the baseline methods as described in Section 5 (since there is no difference between observed data and factual data in this scenario, we have no ICA baseline here). For the CR method, we set the weight of the fairness regularization term as 0.05. The 3000 synthetic data were split into a training set and a test set with a ratio of 80% - 20%. Then we trained a linear

Table 7: Linear regression results on the synthetic datal

Method	MSE (L)	TE	TE_0	TE_1
UF	0.0172 ± 0.0009	1.2499 ± 0.0093	1.2460 ± 0.0252	1.2543 ± 0.0310
CA	0.3467 ± 0.0208	0.5372 ± 0.0057	0.5409 ± 0.0125	0.5316 ± 0.0160
CE	0.7868 ± 0.0556	0.000 ± 0.0000	0.000 ± 0.0000	0.000 ± 0.0000
CR	0.8598 ± 0.0521	0.2572 ± 0.0036	0.2590 ± 0.0066	0.2544 ± 0.0078
Ours	0.4739 ± 0.0202	0.000 ± 0.0000	0.000 ± 0.0000	0.000 ± 0.0000

regression model with the training data and calculate MSE, TE, TE_0, TE_1 with the test data. For each method, we run the experiments five times with different random splits. Table 7 provided the results for the simulation. With the ground truth of the causal model, our proposed method could achieve a 100% counterfactual fairness as the CE method. However, with the use of X, we improved the MSE to a large extent, almost as well as the CA method.

D Detailed Experimental Setup on Real Data

We conducted our experiments using a supercomputing platform. The CPUs used were Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz, and the GPU model was a Tesla V100. Our primary software environments were Python 3.9, Pytorch 1.12.1, and CUDA 10.2.

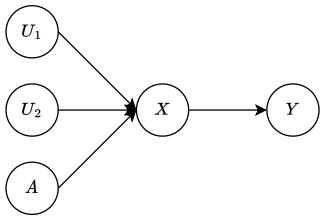

Figure 6: Synthetic causal graph

The VAE structure for the CVAE model is shown in Figure 7. The details for training the VAE can be found in [37]. However, we briefly discuss how the training was done. An encoder takes $[X_\alpha, X_\beta, Y, A]$ as input to generate the hidden variable U. The decoders then serve as structural functions. Decoder f_α takes U as input to generate $\check{X}_\alpha$, decoder f_β takes $[U, \check{A}]$ to generate $\check{X}_\beta$, and f_Y also uses $[U, \check{A}]$ to generate $\check{Y}$.

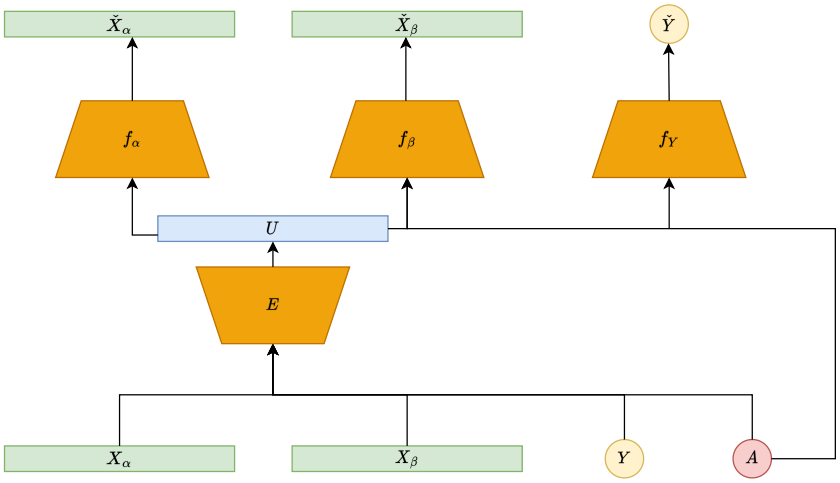

Figure 7: VAE Structure for CVAE Model

During the training of the VAE, we used the following loss function:

$$L = w_\alpha l_\alpha(X_\alpha, \check{X}\alpha) + w_\beta l_\beta(X_\beta, \check{X}\beta) + w_Y l_Y(Y, \check{Y}) + w_u \mathrm{KL}(U||U_p) + W_{fair}||\check{Y}^{[0]} - \check{Y}^{[1]}||_2$$

For the Law School Success dataset, l_α is the BCE loss function, and l_β and l_Y are the MSE loss function. For the UCI Adult Income dataset, l_α, l_β, and l_Y are BCE loss functions. We set $w_\alpha = 1, w_\beta = 1, w_Y = 1, w_u = 1$, and $w_{fair} = 0.15$. The batch size was set to 256 and the learning rate to 0.001. The experiments for the UF, CA, ICA, and CR methods were based on the same VAE.

For the CE and our method, as we needed to use the VAE during the test time, we removed the use of Y from the structure, including the decoder f_Y. Hence, the encoder uses $[X_\alpha, X_\beta, A]$ to obtain U, and $\check{X}_\alpha = f_\alpha(U)$, $\check{X}_\beta = f_\beta(U, \check{A})$. In this case, the loss function becomes:

$$L = w_\alpha l_\alpha(X_\alpha, \check{X}\alpha) + w_\beta l_\beta(X_\beta, \check{X}_\beta) + w_u \text{KL}(U||U_p)$$

For the Law School dataset, we kept the hyperparameters the same, so $w_\alpha = 1, w_\beta = 1, w_u = 1$. For the UCI Adult Income dataset, we set w_α to 1, w_β to 1, and w_u to 1.

Figure 8 depicts the VAE structure for the DCEVAE model. The details for training the VAE can be found in [24], and we summarize it here. The hidden variable is divided into two parts, U_α and U_β. Hence, $U_\alpha = E_\alpha(X_\alpha, Y)$ and $U_\beta = E_\beta(X_\beta, A, Y)$. During the decoding stage, $\check{X}_\alpha = f_\alpha(U_\alpha)$, $\check{X}_\beta = f_\beta(U_\beta, \check{A})$, and $\check{Y} = f_Y(U_\alpha, U_\beta, \check{A})$. A discriminator, D_ψ, is also employed to aid in disentangling U_α and U_β.

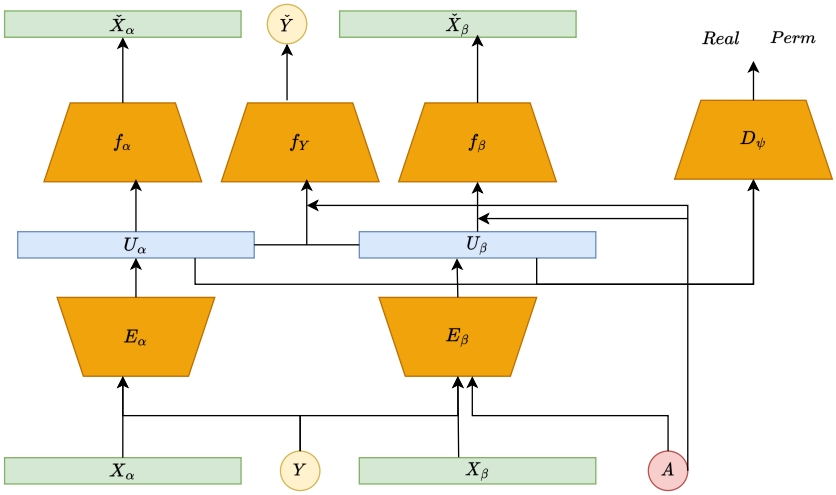

Figure 8: VAE Structure for DCEVAE Model

The training of this VAE can be divided into two stages. In the first stage, we permuted the U_β generated in the batch of data and concatenated them with U_α. The discriminator was trained to distinguish whether a $[U_\alpha, U_\beta]$ is randomly permuted. In the second stage, we trained the encoders and decoders. The loss function is:

$$\begin{aligned} L \quad = \quad & w_\alpha l_\alpha(X_\alpha, \check{X}_\alpha) + w_\beta l_\beta(X_\beta, \check{X}_\beta) + w_Y l_\alpha(Y, \check{Y}) + w_u \text{KL}(U||U_p) \\ & + W_{fair} ||\check{Y}^{[0]} - \check{Y}^{[1]}||_2 \; + w_h * \text{TC} \end{aligned}$$

Here, TC refers to the total correlation loss, which is the negative discrimination loss of D_ψ[7]. We used the same l_α, l_β, l_Y, and weights (w_α, w_β, w_Y) as those used for training the CVAE. We also used the same batch size and learning rate. w_h is set at 0.4, and w_{fair} is set at 0.2. As before, we used the same VAE for the implementation of UF, CA, ICA, and CR methods.

For the CE and our method, we removed all structures related to Y, as we did with the CVAE. For the Law School Success dataset, we kept the hyperparameters the same. And for the UCI Adult Income dataset, we set w_u to 0.5 and w_h to 0.4.

For the finding predictors, we used the linear regression model for the Law School Success Dataset and the logistic regression model for the UCI Adult Income dataset. When training the CR model, we set the coefficient of the regularization term as 0.002.

We split each dataset into a training set, validation set, and test set with a ratio of 60%-20%-20%. The validation set was used to stop the training of the VAE early. The training and validation sets were used together to train the predictors. All experiments were repeated five times with different splits to ensure the results are stable.

[7]More details about the function can be found in [24]

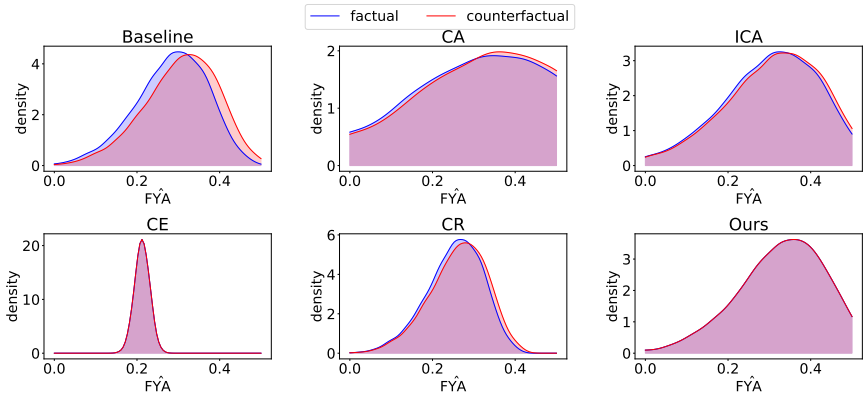

Figure 9: Density distribution of $\hat{\text{FYA}}$ with CVAE causal model

Figure 9 visualizes the PDF of the predicted FYA under the CVAE causal model. As seen in Figure 5, our model is more effective in maintaining the model's behavior for both factual and counterfactual data.

Table 8: Linear regression results on Law School Success dataset with CVAE causal model

Method	MSE (G)	TE (G)	MSE (L)	TE (L)
UF	**0.8664 ± 0.0060**	0.1331 ± 0.0034	**0.8664 ± 0.0060**	0.1258 ± 0.0039
CA	0.8889 ± 0.0097	0.2330 ± 0.0126	0.8915 ± 0.0098	0.2358 ± 0.0127
ICA	0.8704 ± 0.0042	0.1633 ± 0.0014	0.8683 ± 0.0065	0.1543 ± 0.0044
CE	0.8900 ± 0.0076	–	0.8900 ± 0.0076	–
CR	0.8693 ± 0.0064	0.1035 ± 0.0027	0.8696 ± 0.0063	0.0880 ± 0.0025
Ours	0.8689 ± 0.0059	**0.0663 ± 0.0019**	0.8682 ± 0.0060	**0.0655 ± 0.0019**

Table 9: Linear regression results on Law School Success dataset with DCEVAE causal model

Method	MSE (G)	TE (G)	MSE (L)	TE (L)
UF	**0.8677 ± 0.0043**	0.0780 ± 0.0086	**0.8677 ± 0.0043**	0.1300 ± 0.0053
CA	0.8748 ± 0.0050	0.1151 ± 0.00277	0.8794 ± 0.0010	0.1736 ± 0.0398
ICA	0.8687 ± 0.0046	0.0934 ± 0.0160	0.8696 ± 0.0047	0.1372 ± 0.0166
CE	0.8781 ± 0.0068	–	0.8781 ± 0.0068	–
CR	0.8708 ± 0.0042	**0.0463 ± 0.0049**	0.8712 ± 0.0053	**0.0821 ± 0.0052**
Ours	0.8679 ± 0.0045	0.0693 ± 0.0037	0.8692 ± 0.0047	0.0968 ± 0.0024

Tables 8 and 9 present the results for the Law School Success dataset under path-dependent counterfactuals. In these tables, MSE(L) and TE(L) represent the MSE and TE when the LSAT is not in any unfair path, while MSE(G) and TE(G) correspond to the scenario in which GPA is not in any unfair path. The results affirm that our method consistently satisfies PCF in every case. Although the CR method can achieve PCF similar to our method when GPA or LSAT is not in any unfair path of the DCEVAE causal model, it fails in other scenarios because it does not guarantee counterfactual fairness.