Counterfactual Augmentation for Multimodal Learning Under Presentation Bias

Victoria Lin*
Carnegie Mellon University
vlin2@andrew.cmu.edu

Louis-Philippe Morency
Carnegie Mellon University
morency@cs.cmu.edu

Dimitrios Dimitriadis
Microsoft Research
didimit@microsoft.com

Srinagesh Sharma
Microsoft
srsharm@microsoft.com

Abstract

In real-world machine learning systems, labels are often derived from user behaviors that the system wishes to encourage. Over time, new models must be trained as new training examples and features become available. However, feedback loops between users and models can bias future user behavior, inducing a *presentation bias* in the labels that compromises the ability to train new models. In this paper, we propose *counterfactual augmentation*, a novel causal method for correcting presentation bias using generated counterfactual labels. Our empirical evaluations demonstrate that counterfactual augmentation yields better downstream performance compared to both uncorrected models and existing bias-correction methods. Model analyses further indicate that the generated counterfactuals align closely with true counterfactuals in an oracle setting.

1 Introduction

Deployment of machine learning models is ubiquitous in the real world, ranging from web search ranking to movie recommendation. To ensure good performance, new models must be trained periodically, since new training examples may become available, and the types of features that are collected can evolve over time (e.g., from tabular to multimodal data). For user-facing models like recommenders, labels are often derived from user behaviors that the model wishes to encourage, and user-model interactions continuously produce new data that can be used for training models. Modern NLP, for instance, relies heavily on models that learn from user feedback, not the least of which are ChatGPT and other large language models that comprise the current state-of-the-art.

In practice, however, feedback loops between the user and the model can influence future user behavior, inducing *presentation bias* over the labels (Joachims et al., 2017; Pan et al., 2021). By shifting the label distribution away from users' true preferences, presentation bias compromises the ability to train new models (Schmit and Riquelme, 2018; Krauth et al., 2020). For example, an algorithm may present future content based on a user's interactions with prior content. As the user engages with the algorithm's recommendations or outputs, it will recommend more of that same type of content—even if there are other types of content the user might also enjoy (Figure 1).

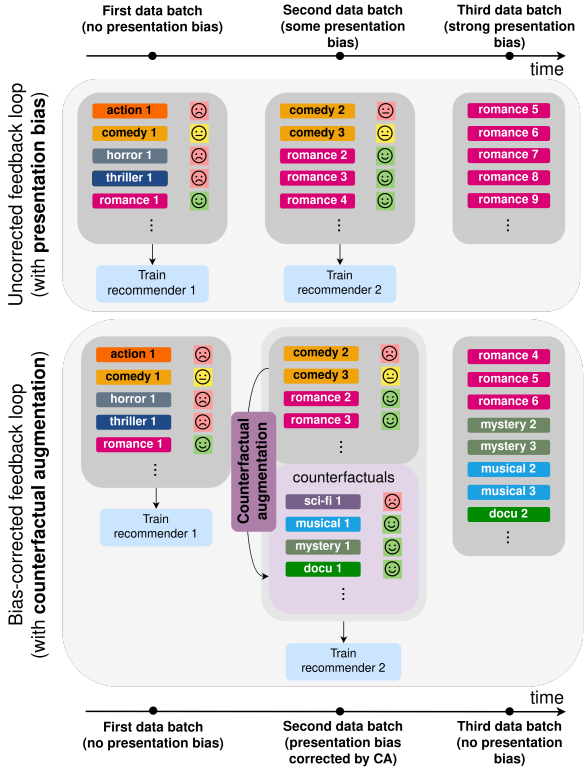

Figure 1: An illustration of how presentation bias may arise from feedback loops (e.g., in a movie recommendation system). The top sequence depicts uncorrected presentation bias, while the bottom sequence demonstrates how our method, *counterfactual augmentation*, can correct presentation bias.

*The majority of this work was conducted while this author was an intern at Microsoft.

Findings of the Association for Computational Linguistics: EMNLP 2023, pages 592–606
December 6-10, 2023

Presentation bias negatively affects the data distribution in two major ways. (1) *Bias amplification.* New labels are dependent on the prior behavior of the model, so they may not reflect the user's true preferences. This bias will amplify as more training loops are completed on biased data. (2) *Label homogenization.* As the model learns user behaviors, most users' responses to its recommendations will be positive, so variation in user feedback decreases.

In this paper, we aim to correct the presentation bias resulting from feedback loops. We first propose that presentation bias arises due to the causal relationship between a model's recommendations and a user's behavior, which affects *which labels are observed*. Users tend to interact with recommended items, so under presentation bias, we are more likely to observe labels for recommended items—while without presentation bias, users would interact with all items (or a random subset), so labels would be observed for the full distribution. We conclude that we can break the causal link behind presentation bias with a *counterfactual* question: how would users have reacted *had they interacted with all items*, contrary to reality?

With this idea as our foundation, we introduce *counterfactual augmentation* (Figure 1), a causal approach for reducing presentation bias using generated counterfactual labels. [1] Because "true" counterfactuals are by definition unknown, counterfactual augmentation leverages the causal relationship between the model's behavior and the user's behavior to generate realistic counterfactual labels. We generate counterfactuals for the labels that are unobserved due to presentation bias, then augment the observed labels with the generated ones. Intuitively, this supplies labels over the full data distribution, yielding a bias-corrected dataset.

We evaluate our method on predictive tasks in language and multimodal settings that reflect real-world presentation bias. We consider data with evolving feature spaces, where over time the features transition from simpler features to richer language or multimodal ones.[2] In our experiments, we demonstrate that counterfactual augmentation effectively corrects presentation bias when training predictive models, outperforming both uncorrected models and existing bias correction methods. We conduct model analyses that examine why counterfactual augmentation is effective for reducing presentation bias and discover that our generated counterfactuals align closely with true counterfactuals in an oracle setting.

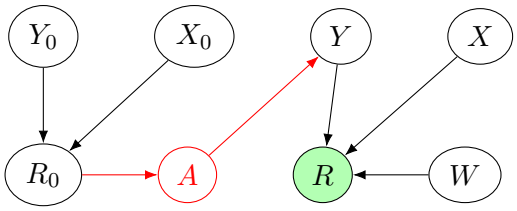

Figure 2: Proposed mechanism of presentation bias. X_t, W_t, and Y_t denote simple features, rich features, and labels at time t (no subscript for $t = 1$), while R_t is a model (e.g., a recommender) trained over the input features and labels. A indicates which items a user interacts with.

2 Problem Statement

We formalize the problem of presentation bias in machine learning systems in causal terms (Figure 2). These systems usually consume both simple features, such as metadata, and *rich* features, such as text or images, training on user interactions with different items to produce recommendations.

Let t be a time index, and let X_t denote the simple features defined over the feature space $\mathbf{X}$. Similarly, let W_t denote rich features defined over the feature space $\mathbf{W}$. We denote *true* user item preferences as $Y_t \in \mathbf{Y}$ and *predicted* user item preferences as R (for simplicity, we can think of these as binary recommendations). Finally, let A_t be an indicator of which items the user interacts with. Due to feature evolution, only X_t is observed at earlier time points, while later both X_t and W_t are observed. Without loss of generality, assume that the feature evolution occurs in the first two time points $t = 0$ and $t = 1$. For ease of notation, we do not include time index subscripts when $t = 1$.

At $t = 0$, a predictive model R_0 is trained on an observed feature set X_0 and labels Y_0. This model makes predictions about user preferences for unseen items and recommends items to the user. These recommendations influence A—which of those items the user subsequently interacts with—because users are much more likely to interact with recommended items, such that $P(A = 1|R_0 = 0) \ll P(A = 1|R_0 = 1)$. In turn, this induces a presentation bias in the distribution of observed Y,

[1] Our code is publicly available at https://github.com/microsoft/causaltransfer.

[2] Although we choose this setting to more accurately represent real-world data, counterfactual augmentation does not require an evolving feature space. In the appendix (Section B.1), we empirically show the effectiveness of counterfactual augmentation in data settings without feature evolution.

the user's measured preferences at $t = 1$. Due to the presentation bias, there is a very high probability of observing Y when $R_0 = 1$ and a very low probability of observing Y when $R_0 = 0$.

At $t = 1$, a full set of simple and rich features (X, W) is observed due to feature evolution. However, because the distribution of Y has been influenced by R_0, a second model R trained on X, W, and Y will not correctly learn user preferences.

Example. Consider a system that must categorize a user's emails as important and unimportant. X is email metadata, and W is the text of the email. Y is an indicator of whether the user interacted with the email positively (e.g., replied) or negatively (e.g., reported spam). R_0 is a classifier trained on X_0 and Y_0 to label emails as important or unimportant. Users preferentially interact with important emails, so emails with $R_0 = 1$ have a much higher chance of user interaction (i.e., $A = 1$) and therefore of having an observed label Y, inducing a bias in Y that depends on R_0. After R_0 is trained, the system's administrators want to train a new, improved model R using both X and W. However, the bias in Y will affect the ability to train R.

3 Methods

To eliminate presentation bias, we notice that we must block the causal path between R_0 and Y so that R_0 no longer influences which Y are observed. Because these two variables are linked by the *mediator* A, we can block the path by controlling for A. To do so, we define the *counterfactual* $Y^{A=a}$, the value Y would have taken *had* $A = a$.

3.1 Counterfactual augmentation

Using $Y^{A=a}$, we block the path between the recommender R_0 and the label Y with the following intuition. A indicates which items users interact with and thus which labels are observed. We can therefore eliminate the influence of A by generating a synthetic data distribution in which *all* items receive user interaction and *all* Y are "observed."

Formally, let $P(Y)$ denote the marginal distribution of the labels and $P(X, Y)$ denote the joint marginal distribution of the features and the labels. In an unbiased setting, a model f is optimized over data $(x, y) \sim P(X, Y)$. Under presentation bias, however, only a portion of $P(Y)$ is observed: the conditional distribution $P(Y|A = 1)$. Consequently, the model f is trained over data $(x, y) \sim P(X, Y|A = 1)$, which may lead to convergence to a non-optimal solution.

Definition 3.1 (Counterfactual augmentation)**.** To correct presentation bias in the data distribution, counterfactual augmentation creates an approximation of the marginal label distribution $P(Y)$ using the estimated distribution of *counterfactual* labels $Y^{A=1}$, or what Y would have been had $A = 1$. This allows us to define $P_{CA}(Y)$, a counterfactually augmented marginal label distribution:

$$P_{CA}(Y) = P(Y|A = 1)P(A = 1) + \underbrace{\hat{P}(Y^{A=1}|A = 0)P(A = 0)}_{\text{counterfactual augmentation}}$$

Combining labels from $P_{CA}(Y)$ with the known features, we have $P_{CA}(X, Y)$, a counterfactually augmented marginal data distribution:

$$P_{CA}(X, Y) = P(X, Y|A = 1)P(A = 1) + \underbrace{P(X, \hat{Y}^{A=1}|A = 0)P(A = 0)}_{\text{counterfactual augmentation}}$$

From $P_{CA}(X, Y)$, bias-corrected data can be sampled, such that the model f is now optimized over $(x, y) \sim P_{CA}(X, Y)$. Supposing $P_{CA}(X, Y)$ is a good approximation of $P(X, Y)$, f should converge to a near-optimal solution.

3.2 Multimodal counterfactual GAN

We implement counterfactual augmentation with a generative adversarial network (GAN) capable of generating realistic counterfactual labels given multimodal input data.[3] Inspired by the work of Yoon et al. (2018), who propose a GAN (GANITE) specifically for estimating individual causal effects, we generate labels—both factual and counterfactual—with a generator G, then train a discriminator D to distinguish factual from counterfactual labels. Our architecture (Figure 3) extends their work in several core aspects:

Mediators. Rather than estimating the direct effect of an intervention A on an outcome Y, we seek to model the indirect effect of a variable R on an outcome Y through the *mediator* A. We account for both of these dependencies, allowing us to later block the effect of R on Y by intervening on A.

[3] We note that the main novelty of our paper is the principle of counterfactual augmentation for correcting presentation bias, rather than the particular means by which the counterfactual is estimated. Empirically, however, we found our counterfactual GAN implementation to generalize well.

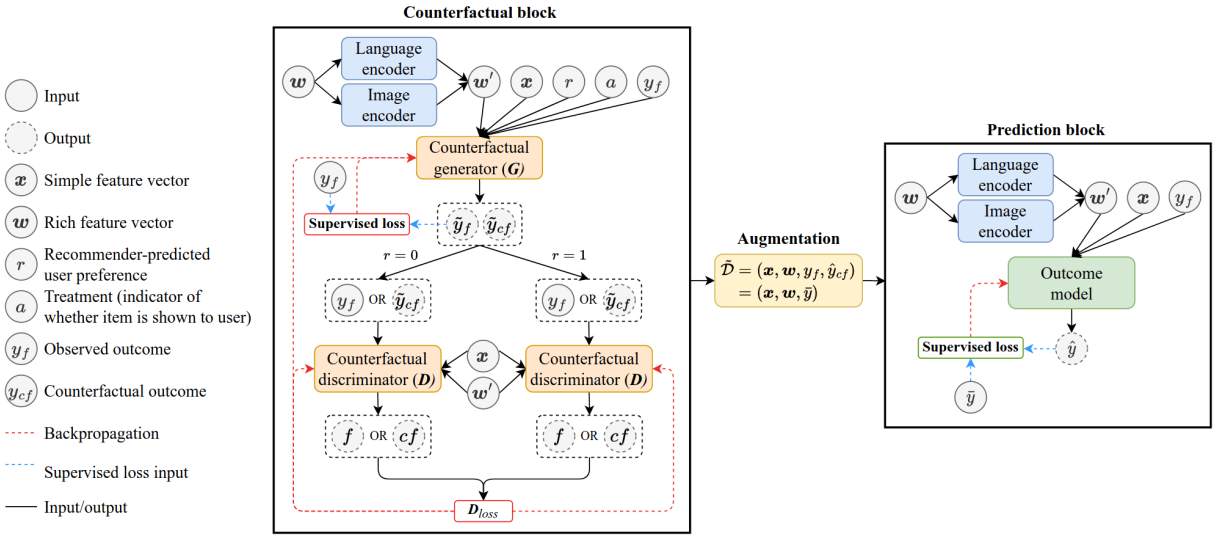

Figure 3: Diagram of our multimodal counterfactual GAN architecture. In the *counterfactual block*, a generator G takes multimodal data as input and generates a factual label $\tilde{y}_f$ and a counterfactual label $\tilde{y}_{cf}$. As the true factual label y_f is known, it is used to learn a supervised loss between y_f and $\tilde{y}_f$ that helps to train G. At random, either the true factual label y_f or the generated counterfactual label $\tilde{y}_{cf}$ is passed to a discriminator D, conditional on the recommendation r corresponding to the label. The discriminator must determine whether the label it has received is factual or counterfactual, and its loss D_{loss} is used to further train both G and D. After the GAN has been trained, its counterfactuals are used to augment data that is used for predictive tasks (e.g., *prediction block*).

Language and multimodal data. Where GANITE was designed for tabular data only, our implementation handles richer features like text and images. We integrate language and image encoders into the architecture that can be simultaneously fine-tuned as the counterfactual GAN is trained.

Correcting the discriminator constraint. Let Y_{cf} denote a counterfactual label and Y_f denote a factual label. The discriminator of GANITE encourages $\hat{P}(Y_{cf}|X) \rightarrow P(Y_f|X)$, i.e., the estimated distribution of counterfactual labels should converge to the true distribution of factual labels. However, in feedback loops, the label is much more likely to be observed (i.e., factual) when $R = 1$ than when $R = 0$. Then the discriminator may enforce $\hat{P}(Y_{cf}|X, R = 0) \rightarrow P(Y_f|X, R = 1)$, which would mean that labels follow the same distribution regardless of whether $R = 1$ or $R = 0$.

We address this problem by defining two separate discriminators—one for each recommendation condition. Each discriminator is arbitrarily passed a factual or counterfactual label from its recommendation condition, and it must identify whether the label is factual or counterfactual. The separate discriminators encourage the realistic constraints $\hat{P}(Y_{cf}|X, R = 0) \rightarrow P(Y_f|X, R = 0)$ and $\hat{P}(Y_{cf}|X, R = 1) \rightarrow P(Y_f|X, R = 1)$.

4 Experiments

We conduct empirical evaluations to assess how well counterfactual augmentation corrects presentation bias, with the aim of improving downstream performance. We evaluate on predictive machine learning tasks, which reflect real-world models' goals of predicting user behavior, and in multiple data settings, both synthetic and real-world. To facilitate detailed analysis of our models, we introduce a procedure for inducing realistic presentation bias in unbiased datasets. All data and code for our experiments will be publicly released.

4.1 Datasets

To recreate feature evolution in our experiments, we evaluate on datasets that contain both tabular features and rich features like text or images. We select two datasets from the Multimodal AutoML Benchmark (Shi et al., 2021): Airbnb and Clothing. **Airbnb** consists of 22,895 Airbnb listings for the city of Melbourne, including metadata, text descriptions, and images of the property. The nightly price of the listing is the label. **Clothing** comprises 23,486 reviews of women's clothing from an online retailer, with metadata, the title, and the text of the review. The review score is treated as the label. Both labels can be predicted directly via regression, but they can also be discretized to be used in clas-

sification tasks (as we do in our evaluation). For our binary classification tasks, we binarize both datasets in a 0-1 proportion of approximately 0.25 to 0.75 to reflect real-world data, in which the majority of feedback received from users is positive.

We further create a synthetic version of the Airbnb dataset (**Synthetic**) in which the features are taken from the real dataset, but the label is synthesized as a noisy function of the tabular features and the multimodal features. The purpose of this dataset is to evaluate the efficacy of counterfactual augmentation in a "best-case scenario" in which we know that there is some signal about the label that can be gained independently from both the simple features and the rich features. We use a binary label, again to reflect a "best-case scenario" in which the downstream task is relatively easy (compared to multi-class classification or regression), with a 0-1 proportion of approximately 0.25 to 0.75.

Additional details about the datasets are provided in the appendix (Section A.1).

4.2 Method for inducing presentation bias

To induce presentation bias in these datasets in a way that will allow for post-hoc model analysis, we use a procedure that mimics feedback loops in real-world systems. We first create three splits of the data, which correspond to the three data batches in Figure 1. We refer to these splits as $D_{original}$, D_{train}, and D_{eval}. On $D_{original}$, which has no presentation bias, we fit a model M_{tab} on the labels $Y_{original}$, using only tabular features.

We use M_{tab} to predict labels R_{train} for D_{train} using tabular features, where $R_{train} = M_{tab}(X_{train})$. R_{train} corresponds to R_0 in our causal structure. Next, we drop 90% of the labels from samples in D_{train} where $R_{train} = 0$ (where Y is multi-class or binary, we use a threshold value instead). This induces presentation bias by creating the causal dependency $R_0 \rightarrow A \rightarrow Y$, where labels are observed with high probability when $R_0 = 1$ and with low probability when $R_0 = 0$. We also randomly drop $\sim$35% of samples from D_{train} with equal probability (reflecting the remaining items that users do not interact with).

Finally, for D_{eval}, we create an **unbiased** version in which we leave D_{eval} as it is, and a **biased** version D_{biased}. For D_{biased}, we again use M_{tab} to predict labels R_{eval} using only tabular features, where $R_{eval} = M_{tab}(X_{eval})$. We then drop 90% of the samples in D_{biased} where $R_{eval} = 0$.

4.3 Models

Baselines. We compare counterfactual augmentation against several baselines. First, we include a model without bias correction (**Uncorrected**). To provide the best chance of achieving good performance, we use pre-trained transformer architectures fine-tuned on the respective task datasets: DistilBERT (Sanh et al., 2020) for language and ViT (Dosovitskiy et al., 2021) for images. These models are used as encoders for the text and images of the datasets. Once embeddings are obtained, they are concatenated with the tabular data and passed to a final layer fine-tuned for the predictive task.

Our remaining baselines are implementations of existing methods for correcting presentation bias, both of which we describe further in Section 6. In our experiments, the **IPW** baseline is implemented identically to the uncorrected baseline; however, when fine-tuning the final task layer, an *inverse propensity weighted* loss (Wang et al., 2016) is used. The **Dragonnet** baseline is an adaptation of a method proposed by Shi et al. (2019) for jointly estimating causal treatments and outcomes with a single neural network. To make this method compatible with our data setting, we pre-embed the text and images before passing them to Dragonnet, and we also modify the final layer to output estimated counterfactuals rather than estimated causal effects.

Counterfactual augmentation. In our proposed method, counterfactual augmentation (**CA**), we train our multimodal counterfactual GAN on a biased dataset, then use the GAN to generate the counterfactual labels for all samples for which labels are not observed. Combining the generated labels with the observed labels, we have a bias-corrected dataset. With this bias-corrected data, we encode text and images using fine-tuned DistilBERT and ViT, combine them with the tabular data, and train a final layer for the specific task.

Additional details about the training procedures are provided in the appendix (Section A.2).

4.4 Evaluation

In our evaluation, our models are fit on D_{train} (which contains presentation bias) and evaluated on both D_{eval} and D_{biased}. Evaluation on D_{eval} indicates how well our model predicts the label in a setting where presentation bias is not a factor (i.e., if we knew all labels). Evaluation on D_{biased} indicates how well our model predicts the label given the data that is available to us in reality.

	Synthetic				Airbnb				Clothing			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	81.7	76.9	58.4	27.2	84.8	82.5	69.5	48.0	77.5	68.1	45.5	3.8
IPW	82.0	78.4	62.0	34.4	86.0	84.7	74.2	56.8	79.6	73.8	56.8	25.4
Dragonnet	81.9	79.8	65.9	42.5	81.8	79.9	65.7	42.1	77.1	67.1	43.5	0.0
CA (ours)	**82.7**	**82.0**	**71.1**	**52.8**	**87.4**	**87.4**	**80.2**	**68.2**	**80.3**	**76.7**	**63.1**	**37.9**
Improvement	0.7%	2.2%	5.2%	10.3%	1.5%	2.7%	6.0%	11.5%	0.7%	2.9%	6.3%	12.5%
Oracle	82.6	79.7	64.8	39.7	88.5	88.4	81.8	70.7	80.1	74.6	58.3	28.2

Table 1: Results on binary classification tasks (unbiased evaluation dataset).

	Synthetic				Airbnb				Clothing			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	**85.1**	80.5	54.9	17.9	88.2	86.1	67.1	40.8	79.9	71.3	45.7	2.5
IPW	85.0	81.0	56.9	22.0	88.5	87.1	70.7	47.8	**81.4**	**75.7**	**55.5**	**21.5**
Dragonnet	81.0	79.3	57.1	25.0	86.4	81.1	51.4	10.2	79.6	70.6	44.3	0.0
CA (ours)	84.9	**83.0**	**64.0**	**36.6**	**88.7**	**88.2**	**74.5**	**55.6**	80.5	74.6	53.6	18.2
Improvement	-	2.0%	6.9%	11.6%	0.2%	1.1%	3.9%	7.8%	-	-	-	-
Oracle	85.1	81.6	58.8	25.9	89.7	89.5	77.9	61.7	81.8	76.3	56.8	24.0

Table 2: Results on binary classification tasks (biased evaluation dataset).

	Regression				Multi-class		
	Airbnb		Clothing		Clothing		
	R^2	RMSE	R^2	RMSE	Acc.	F_1	F_{1mac}
Uncorrected	0.126	0.935	0.092	0.953	55.7	39.9	14.4
IPW	0.127	0.934	0.120	0.938	56.0	40.9	15.7
Dragonnet	-	-	-	-	-	-	-
CA (ours)	**0.186**	**0.902**	**0.197**	**0.896**	**57.1**	**44.6**	**20.1**
Improvement	5.9%	3.2%	7.6%	4.1%	1.1%	3.7%	4.4%
Oracle	0.131	0.932	0.194	0.898	57.2	44.3	19.2

Table 3: Results on regression and multi-class classification tasks (unbiased evaluation dataset). Our RMSE metric is normalized RMSE, or RMSE divided by the standard deviation of the evaluation set.

	Regression				Multi-class		
	Airbnb		Clothing		Clothing		
	R^2	RMSE	R^2	RMSE	Acc.	F_1	F_{1mac}
Uncorrected	0.112	0.942	0.083	0.958	58.4	43.2	14.9
IPW	0.112	0.942	0.098	0.950	**58.6**	43.8	15.8
Dragonnet	-	-	-	-	-	-	-
CA (ours)	**0.134**	**0.930**	**0.128**	**0.934**	58.6	**43.9**	**16.1**
Improvement	2.2%	1.2%	3.0%	1.6%	-	0.1%	0.3%
Oracle	0.108	0.944	0.156	0.919	59.3	46.0	18.2

Table 4: Results on regression and multi-class classification tasks (biased evaluation dataset).

We note that as a consequence of presentation bias, any class or distribution imbalance in the labels Y will be amplified, since there is a positive relationship between the predicted labels R and the true labels Y. This imbalance reflects the real-world tendency for users to like their recommendations and for positive labels to dominate. Therefore, in classification tasks, overall accuracy and F_1 score will be artificially high for a model that simply predicts the most common class. Important measures of success for a method will instead be $F_{1_{mac}}$, or macro F_1 score (F_1 score uniformly weighted across all classes), and for binary classification, $F_{1_{min}}$, or F_1 score on the minority class.

5 Results and Discussion

5.1 Prediction task results

We evaluate counterfactual augmentation against our baselines in the Synthetic, Airbnb, and Clothing data settings on binary classification tasks (Tables 1 and 2) and multi-class classification and regression tasks (Tables 3 and 4). "Improvement" is computed by taking the difference between the CA score and the score of the next-best method for that metric, which is generally IPW.

We observe that when evaluating in an unbiased setting (which reflects "true" preferences), counterfactual augmentation offers the best performance across all metrics for all tasks on all datasets, often by a significant margin. It outperforms not only the uncorrected baseline but also both bias-correction baselines, IPW and Dragonnet. When evaluating in a biased setting (which reflects the evaluation data we have available in reality), counterfactual augmentation also improves performance across

metrics, tasks, and datasets. In general, it outperforms competing bias-correction methods, with the single exception being the binary classification task for the Clothing Review dataset, where it does not achieve as much improvement as IPW but still offers substantial gains over the uncorrected model.

Importantly, the biggest improvements resulting from counterfactual augmentation are in the minority classes. As we mention previously, due to the imbalance in the distribution of Y, macro and minority class F_1 score are the best measures of performance. Furthermore, since the generated counterfactual labels correspond largely to the minority classes, the relatively high minority class F_1 scores suggest that the generated counterfactuals are sufficiently realistic to allow the model to learn.

Taken together, these results suggest that *counterfactual augmentation is indeed successful in correcting presentation bias—and that it does a better job than existing bias-correction methods*. From an empirical standpoint, counterfactual augmentation is both useful and stable across settings and tasks, yielding consistently good performance.

5.2 Model analysis: Why does counterfactual augmentation work?

Counterfactual augmentation produces clear empirical gains in downstream performance over both uncorrected models and existing bias-correction methods. In this section, we analyze our generated counterfactuals to better understand these improvements. Although true counterfactuals are never known in the real world, in these experiments we do have access to the true counterfactuals of D_{train}, which we withheld in the process of inducing presentation bias. We use these as a basis for comparison with our generated counterfactuals.

Comparing counterfactual distributions. To assess how well our generated counterfactuals correspond to real counterfactuals, we plot their distributions together for each combination of dataset and task (Figure 4). We observe that for the easier binary classification task, *the distribution of generated counterfactuals closely reflects that of the true counterfactuals across all data settings*. On these tasks, it appears that the generated counterfactuals are a good approximation of the true counterfactuals. For the more difficult multi-class classification and regression tasks, the difference between the generated and true distributions is greater.

Reducing presentation bias helps correct the label imbalance that exists in the overall label distribution (Figure 5). However, the generated counterfactual distribution also tends to be more uniform compared to the true counterfactual distribution (seen in 4 out of 6 plots in Figure 4). Therefore, even aside from the reduced presentation bias, the greater uniformity of the generated counterfactuals may further correct label imbalance. In general, we observe that *the bias-corrected label distribution* $P_{CA}(Y)$ *is more balanced than the uncorrected label distribution* $P(Y|A = 1)$. This reduction in label imbalance better enables a model to learn on the bias-corrected set.

Performance of an oracle. Because we have access to the true counterfactuals, we can train an oracle model over an unbiased version of D_{train}. By comparing the oracle to counterfactual augmentation, we can determine how well counterfactual augmentation recovers performance compared to the original unbiased data. We report the results of the oracle in Tables 1 through 4.

For most tasks and data settings, we observe that—as expected—counterfactual augmentation still results in some loss of performance compared to the oracle.[4] However, the performance gap between CA and the oracle is generally substantially less than the performance gap between CA and the next-best bias-correction method. These results suggest that although CA constitutes a significant improvement over existing methods, further refinement of the counterfactual generation method may be able to yield even better results.

6 Related Work

Presentation bias may be considered a type of *selection bias*, in which the sampling distribution differs from the population distribution. Selection bias is a core challenge of observational causal inference, where the causal effect of a treatment A on an outcome Y is estimated not from a randomized trial but from observed data. Since the treatment assignment mechanism is not random, it must be accounted for during estimation.

One common method for addressing selection bias in causal inference is *inverse propensity*

[4] In some cases, we see that the oracle achieves worse performance than CA. We posit that the tendency of the generated counterfactual distribution toward the uniform "over-corrects" label imbalance, so the bias-corrected label distribution is more uniform than the unbiased label distribution. This may make model training easier on the bias-corrected distribution (e.g., on sparser portions of the unbiased distribution).

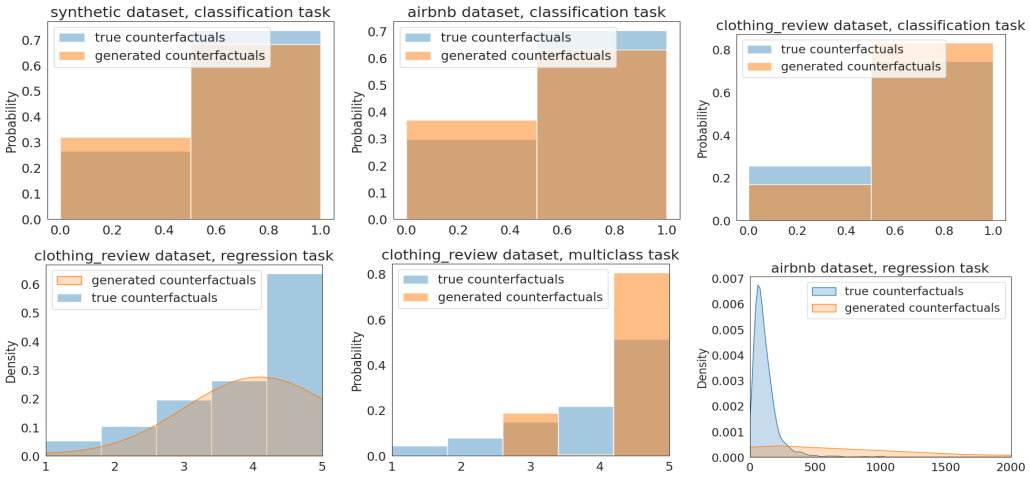

Figure 4: Comparison of the distributions of the generated counterfactuals and the true counterfactuals.

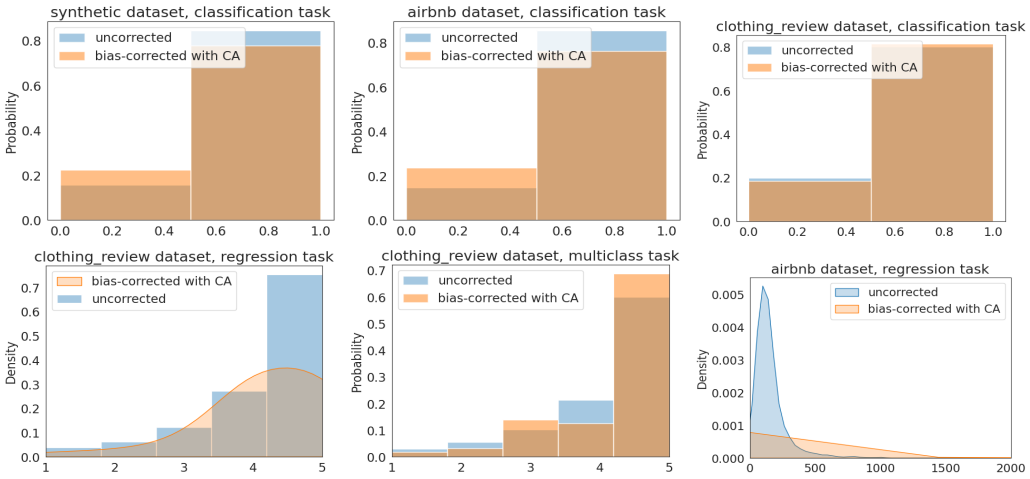

Figure 5: Comparison of uncorrected label distributions and label distributions after bias-correction with CA.

weighting (IPW) (Robins et al., 1994; Hernán and Robins, 2006). At a high level, IPW up-weights samples corresponding to treatment conditions that are unlikely to be observed and down-weights samples corresponding to treatment conditions that are likely to be observed, such that all treatment conditions appear to be equally likely over the data distribution. This blocks the causal path between the treatment assignment and the outcome.

IPW for presentation bias correction. Using this principle, a number of works propose an inverse propensity weighted empirical loss function that can be used to reduce the effects of presentation bias when training a model on biased data (Wang et al., 2016; Schnabel et al., 2016; Joachims et al., 2017). Several works also engage with IPW in more complex ways. Krauth et al. (2022) address a longitudinal bias setting and propose an algorithm that maximizes the desired outcome at each time step using an IPW-based estimator. Shi et al. (2019) introduce Dragonnet, a fully-connected multi-head neural network that jointly predicts the treatment and the outcome, simultaneously yielding both a propensity score estimate and a predicted outcome.

Task-based presentation bias correction. Because presentation bias can appear in many task settings, there exist a number of task-specific approaches for reducing presentation bias. In information retrieval, for example, unbiased learners of click data (Ai et al., 2018) and propensity-weighted rank metrics (Agarwal et al., 2019) have been proposed, while in the recommender literature, methods have been developed for the matrix factorization setting (Bonner and Vasile, 2018; Wang et al., 2020). However, the task-specific nature of these methods limits their generalizability compared to counterfactual augmentation.

Estimating counterfactuals. The inability to

know an individual's counterfactual is a central challenge of causal inference. However, recent works in the deep learning literature have made large inroads toward estimating individual treatment effects (Shalit et al., 2017; Louizos et al., 2017; Yoon et al., 2018), which is an adjacent task to estimating individual counterfactuals. We draw upon this body of work as a basis for obtaining high-quality counterfactuals.

Counterfactuals in NLP. Our work is contextualized within a recent body of research that has shown that counterfactuals are an effective supplement to training data when learning language models (Wang and Culotta, 2021; Qian et al., 2021; Yang et al., 2021; Howard et al., 2022). Existing works largely rely on manually created counterfactuals or programmatically generated counterfactuals. Our method advances beyond prior works by leveraging the causal mechanism behind the missing portions of the data distribution to efficiently generate targeted, high-quality counterfactuals.

7 Conclusion

In this paper, we introduced *counterfactual augmentation*, a causal method for correcting presentation bias using generated counterfactuals. We described the causal mechanism behind presentation bias in real-world machine learning systems that rely on user feedback, and we explained the causal reasoning behind counterfactual augmentation. We presented empirical evaluations using counterfactual augmentation to reduce presentation bias, and we found that our approach significantly outperforms existing methods. Finally, we conducted a model analysis to explore why counterfactual augmentation is effective in addressing presentation bias. Given the prevalence of presentation bias in real-world deployments of machine learning models, our findings suggest that counterfactual augmentation has the potential to improve the quality of user-facing machine learning models across many types of applications.

8 Acknowledgements

This material is based upon work partially supported by Microsoft, the National Science Foundation (awards 1722822 and 1750439), and the National Institutes of Health (awards R01MH125740, R01MH132225, R01MH096951, and R21MH130767). Victoria Lin is partially supported by a Meta Research PhD Fellowship. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors, and no official endorsement should be inferred. We are grateful to Alan Thomas and Sebastian de la Chica for many helpful discussions and feedback.

9 Limitations

The most significant limitation of counterfactual augmentation is its requirement that the generated counterfactuals be sufficiently close to true counterfactuals; otherwise, the counterfactually augmented data distribution $P_{CA}(X, Y)$ is not a good approximation of the true data distribution $P(X, Y)$. If poor-quality counterfactuals are produced and $P_{CA}(X, Y)$ is very different from $P(X, Y)$, counterfactual augmentation could instead hurt models that are trained on the augmented data. Although our multimodal counterfactual GAN generates high-quality counterfactuals for the tasks and data settings that we evaluate, we do not know if this will be the case across every task and data setting. A different counterfactual estimation method may be required depending on the particular problem.

Based on failure modes of causal effect estimation in statistical causal inference, we hypothesize that lower-quality counterfactuals may be produced if:

- The causal mechanism of presentation bias is misspecified.
- The feature data is very noisy or sparse, making it difficult to learn counterfactuals.
- The counterfactual generation model does not have enough capacity to model the data (could be more of a problem for "traditional" statistical linear models).

10 Ethics Statement

Broader impact. Deep learning models have been shown to perpetuate and even amplify the biases in their training data (Bolukbasi et al., 2016; Swinger et al., 2019; Caliskan et al., 2017). Often, these biases manifest in a similar way to presentation bias: that is, only a portion of the theoretical data distribution is contained in the model's training dataset, which impacts what the model learns.

Therefore, we believe that counterfactual augmentation may be helpful not only in correcting presentation bias but also in reducing social biases in data. In principle, counterfactual augmentation can be used to correct any type of bias for which the causal mechanism is known. The causal mechanism is used to generate counterfactuals, which augment the unobserved portion of the data distribution. Consequently, counterfactual augmentation may also be helpful in correcting social biases and helping make data more fair.

Ethical considerations. When used in conjunction with multimodal data, as it is in this paper, counterfactual augmentation relies in part on large pre-trained models to generate counterfactuals. As a result, it is also possible that the generated counterfactuals themselves may encode the biases contained in large pre-trained models. Users should be cautious when employing counterfactual augmentation in sensitive settings or when using it to reduce biases on protected attributes.

Additionally, we acknowledge the environmental impact of large language and image models, which are used in this work.

References

Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 5–14, New York, NY, USA. Association for Computing Machinery.

Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, page 385–394, New York, NY, USA. Association for Computing Machinery.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Stephen Bonner and Flavian Vasile. 2018. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 104–112, New York, NY, USA. Association for Computing Machinery.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

Miguel A Hernán and James M Robins. 2006. Estimating causal effects from epidemiological data. *Journal of Epidemiology & Community Health*, 60(7):578–586.

Phillip Howard, Gadi Singer, Vasudev Lal, Yejin Choi, and Swabha Swayamdipta. 2022. NeuroCounterfactuals: Beyond minimal-edit counterfactuals for richer data augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5056–5072, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, page 781–789, New York, NY, USA. Association for Computing Machinery.

Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I. Jordan. 2020. Do offline metrics predict online performance in recommender systems?

Karl Krauth, Yixin Wang, and Michael I. Jordan. 2022. Breaking feedback loops in recommender systems with causal inference.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. 2017. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Weishen Pan, Sen Cui, Hongyi Wen, Kun Chen, Changshui Zhang, and Fei Wang. 2021. Correcting the user feedback-loop bias for recommendation systems.

Chen Qian, Fuli Feng, Lijie Wen, Chunping Ma, and Pengjun Xie. 2021. Counterfactual inference for text classification debiasing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5434–5445, Online. Association for Computational Linguistics.

James M. Robins, Andrea Rotnitzky, and Lue Ping Zhao. 1994. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, 89(427):846–866.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Sven Schmit and Carlos Riquelme. 2018. Human interaction with recommendation systems. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 862–870. PMLR.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1670–1679, New York, New York, USA. PMLR.

Uri Shalit, Fredrik D. Johansson, and David Sontag. 2017. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3076–3085. PMLR.

Claudia Shi, David Blei, and Victor Veitch. 2019. Adapting neural networks for the estimation of treatment effects. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alexander J. Smola. 2021. Benchmarking multimodal automl for tabular data with text fields. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*.

Nathaniel Swinger, Maria De-Arteaga, Neil Thomas Heffernan IV, Mark DM Leiserson, and Adam Tauman Kalai. 2019. What are the biases in my word embedding? In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 305–311, New York, NY, USA. Association for Computing Machinery.

Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 115–124, New York, NY, USA. Association for Computing Machinery.

Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. 2020. Causal inference for recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 426–431, New York, NY, USA. Association for Computing Machinery.

Zhao Wang and Aron Culotta. 2021. Robustness to spurious correlations in text classification via automatically generated counterfactuals. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14024–14031.

Linyi Yang, Jiazheng Li, Padraig Cunningham, Yue Zhang, Barry Smyth, and Ruihai Dong. 2021. Exploring the efficacy of automatically generated counterfactuals for sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 306–316, Online. Association for Computational Linguistics.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*.

A Experimental Details

A.1 Data

Details of our data splits are provided in Table 5, including dataset composition and licensing information. Both Airbnb and Clothing are publicly available datasets, and the Synthetic dataset is available on our GitHub. This release of the Synthetic dataset—for research purposes only—is compatible with the intended use of the Airbnb dataset, on which it is based. All datasets are in English.

A.2 Training details

Our use of all pre-existing models is limited to research purposes only and is compliant with their intended use.

Language and image models. Our language and image transformer models were built on the HuggingFace[5] `transformers` library (version 4.18.0), with pre-trained models taken from the HuggingFace model hub. For fine-tuning, we used an Adam optimizer and learning rates $[2 \times 10^{-1}, 2 \times 10^{-2}, 2 \times 10^{-3}, 2 \times 10^{-4}, 2 \times 10^{-5}]$, and we found 2×10^{-5} to be the best learning rate across all models. We trained for 5 epochs and selected the model with the best validation loss. All other hyperparameters were set to Trainer class defaults from the `transformers` library.

Dragonnet. Our implementation of Dragonnet was based on its public code release,[6] which uses Tensorflow (version 2.8.0). All hyperparameters were kept to their default values from the original code.

Multimodal counterfactual GAN. Our implementation of our multimodal counterfactual GAN uses PyTorch (version 1.11.0) and was built on

[5] https://huggingface.co/

[6] https://github.com/claudiashi57/dragonnet

Dataset	$D_{original}$	D_{train}	D_{eval}	D_{biased}	n_{total}	License
Synthetic	4,572	5,400	9,135	5,400	22,895	CC0 1.0
Airbnb	4,572	5,400	9,135	5,400	22,895	CC0 1.0
Clothing	4,698	5,400	9,394	5,400	23,486	CC0 1.0

Table 5: Composition of data splits. For each dataset, the number of samples in $D_{original}$, D_{train}, D_{eval}, and D_{biased} is given, along with total samples for each dataset. Licensing information is also provided.

skeleton code from the public code release[7] of GANITE. We tune over the following hyperparameters (best in bold):

- Hidden layer size of generator and discriminator: [**128**, 256, 512, 1024]
- Number of generator iterations: [200, **500**, 1000, 2000]
- Number of discriminator iterations: [5, 6, 7, **8**, 9, 10]
- Learning rate: [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, $\mathbf{10^{-5}}$]
- Separate discriminators: [**yes**, no]
- Scaling data: [yes, **no**[8]]
- Fine-tune encoders and GAN sequentially: [**yes**, no]

The number of parameters and licenses for each of the models is reported in Table 6.

Model	# parameters	License
Dragonnet	474,603	Unknown
DistilBERT	66,955,010	Apache-2.0
ViT	85,800,194	Apache-2.0
Counterfactual GAN	500,601	BSD-3-Clause

Table 6: Number of parameters and license for each model. Note that because the best-performing configuration for the counterfactual GAN trains the language and image encoders and the GAN sequentially, the number of parameters for the counterfactual GAN excludes the encoder parameters. When the encoders are fine-tuned in parallel with the GAN, we consider the number of parameters for the counterfactual GAN to be 153,713,529.

A.3 Runtimes

Counterfactual augmentation is computationally efficient, and its computational overhead is minimal compared to the computation required to learn task models. We provide runtimes of all models and bias-correction methods for each of our evaluation tasks in Tables 7, 8, and 9. For every task, either IPW or counterfactual augmentation is the most efficient bias-correction method.

	Synthetic	
	Runtime	Added time
Uncorrected	1304	-
IPW	1350	46
Dragonnet	1494	190
CA (ours)	1361	57

Table 7: Runtimes (in seconds) of all models and bias-correction methods for the Synthetic binary classification task. "Added time" denotes the additional time the bias-correction method requires relative to the uncorrected model.

A.4 Computing resources

A portion of our experiments were conducted using machines with consumer-level NVIDIA graphics cards. Our remaining experiments were conducted using cloud computing resources. We estimate the number of GPU hours used to be around 150.

B Additional Experiments

B.1 Counterfactual augmentation without feature evolution

In this section, we present results (Tables 10, 11, 12, and 13) that verify that counterfactual augmentation does not require feature evolution to successfully correct presentation bias. We follow the same experimental procedures as in Section 4; however, for all three datasets—Synthetic, Airbnb, and Clothing—we use only tabular features for all data splits $D_{original}$, D_{train}, D_{eval}, and D_{biased}.

We observe that as is the case in our main results, counterfactual augmentation generally produces improvements in overall performance and macro/minority class performance, both relative

[7] https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/ganite

[8] For the clothing multi-class classification task only, scaling was the better option.

	Airbnb			
	Binary classification		Regression	
	Runtime	Added time	Runtime	Added time
Uncorrected	1319	-	1320	-
IPW	1367	48	1431	111
Dragonnet	1423	104	-	-
CA (ours)	1383	64	1334	14

Table 8: Runtimes (in seconds) of all models and bias-correction methods for the Airbnb tasks. "Added time" denotes the additional time the bias-correction method requires relative to the uncorrected model.

	Clothing					
	Binary classification		Multi-class classification		Regression	
	Runtime	Added time	Runtime	Added time	Runtime	Added time
Uncorrected	251	-	246	-	247	-
IPW	272	21	271	25	273	26
Dragonnet	351	100	-	-	-	-
CA (ours)	265	15	260	14	317	71

Table 9: Runtimes (in seconds) of all models and bias-correction methods for the Airbnb tasks. "Added time" denotes the additional time the bias-correction method requires relative to the uncorrected model.

	Synthetic				Airbnb				Clothing			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	79.7	70.8	44.6	0.5	86.7	86.1	77.2	62.5	77.1	67.2	43.8	0.5
IPW	82.0	78.6	62.7	35.8	**86.7**	86.5	78.4	64.9	77.1	67.2	43.8	0.5
Dragonnet	82.2	79.3	63.5	37.5	81.6	79.2	64.6	40.1	80.1	76.5	63.0	37.8
CA (ours)	**82.7**	**81.5**	**69.7**	**49.9**	86.2	**86.6**	**79.4**	**67.5**	**80.3**	**76.7**	**63.0**	**37.9**
Improvement	0.5%	2.2%	6.2%	12.4%	-	0.1%	1.0%	2.6%	0.2%	0.2%	0.0%	0.1%

Table 10: Results on binary classification tasks without feature evolution (unbiased evaluation dataset).

	Synthetic				Airbnb				Clothing			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	84.7	77.7	46.2	0.7	**89.1**	**88.4**	74.6	55.5	79.6	70.7	44.5	0.4
IPW	85.0	80.0	53.5	15.2	88.6	88.4	75.2	56.9	79.6	70.7	44.5	0.4
Dragonnet	84.9	78.9	49.8	7.8	85.7	80.9	51.9	11.6	**80.7**	**74.9**	**54.6**	**20.2**
CA (ours)	**85.0**	**81.7**	**59.2**	**26.7**	88.3	88.2	**75.2**	**57.2**	80.5	74.6	53.6	18.2
Improvement	-	1.7%	5.7%	11.5%	-	-	0.0%	0.3%	-	-	-	-

Table 11: Results on binary classification tasks without feature evolution (biased evaluation dataset).

to the uncorrected baseline and to the competing bias-correction baselines, IPW and Dragonnet.

Performance of both the baselines and counterfactual augmentation is less consistent compared to the feature evolution setting (this is particularly evident in the clothing regression task, where R^2 is negative). We hypothesize that there is not sufficient information encoded in the tabular data to

	Regression				Multi-class		
	Airbnb		Clothing		Clothing		
	R^2	RMSE	R^2	RMSE	Acc.	F_1	$F_{1_{mac}}$
Uncorrected	0.079	0.959	-0.419	1.191	55.6	39.8	14.3
IPW	0.083	0.958	-0.419	1.191	55.6	39.8	14.3
Dragonnet	-	-	-	-	-	-	-
CA (ours)	**0.269**	**0.855**	**-0.133**	**1.064**	**57.2**	**44.6**	**20.2**
Improvement	18.6%	10.3%	28.6%	12.7%	1.6%	4.8%	5.9%

Table 12: Results on regression and multi-class classification tasks without feature evolution (unbiased evaluation dataset). Our RMSE metric is normalized RMSE, or RMSE divided by the standard deviation of the evaluation set.

	Regression				Multi-class		
	Airbnb		Clothing		Clothing		
	R^2	RMSE	R^2	RMSE	Acc.	F_1	$F_{1_{mac}}$
Uncorrected	0.041	0.979	-0.329	1.153	58.4	43.0	14.7
IPW	0.043	0.979	-0.329	1.153	58.4	43.0	14.7
Dragonnet	-	-	-	-	-	-	-
CA (ours)	**0.207**	**0.891**	**-0.130**	**1.063**	**58.6**	**43.9**	**16.1**
Improvement	16.4%	8.8%	19.9%	9.0%	0.2%	0.9%	1.4%

Table 13: Results on regression and multi-class classification tasks without feature evolution (biased evaluation dataset).

learn certain tasks well. Furthermore, it is also more difficult to generate the counterfactuals for counterfactual augmentation, since the GAN now also has access only to tabular data.

B.2 Modality ablations

To further demonstrate the utility of counterfactual augmentation in natural language settings, in this section we report results from experiments with ablated (language-only) versions of the Airbnb and Synthetic datasets (Tables 14, 15, 16). Again, we follow the same experimental procedures as in Section 4, but we make only language features available in D_{train}, D_{eval}, and D_{biased}.

We find that—consistent with our other results—counterfactual augmentation continues to outperform uncorrected models and existing bias-correction methods. Interestingly, we observe that bias correction appears to work better in some data settings given only the text modality, relative to models that have access to both text and images.

	Synthetic				Airbnb			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	79.7	70.8	44.5	0.4	82.9	78.8	61.4	32.6
IPW	81.8	77.5	59.8	30.1	86.3	85.7	76.5	61.3
Dragonnet	80.6	74.8	53.2	17.3	81.5	78.8	63.4	37.6
CA (ours)	**82.2**	**81.0**	**68.9**	**48.6**	**87.4**	**87.7**	**81.2**	**70.3**
Improvement	0.4%	3.5%	9.1%	18.4%	1.1%	2.1%	4.7%	9.0%

Table 14: Results on binary classification tasks with the language modality only (unbiased evaluation dataset).

	Synthetic				Airbnb			
	Acc.	F_1	F_{1mac}	F_{1min}	Acc.	F_1	F_{1mac}	F_{1min}
Uncorrected	84.6	77.6	45.8	0.0	87.4	83.8	59.9	26.7
IPW	**84.9**	79.3	51.0	10.2	88.3	87.5	72.2	51.1
Dragonnet	84.7	78.0	46.8	1.9	86.3	80.9	51.2	9.8
CA (ours)	84.6	**81.3**	**58.4**	**25.4**	**88.6**	**88.6**	**76.4**	**59.5**
Improvement	-	2.0%	7.4%	15.2%	0.3%	1.2%	4.2%	8.5%

Table 15: Results on binary classification tasks with the language modality only (biased evaluation dataset).

	Airbnb (unbiased)		Airbnb (biased)	
	R^2	RMSE	R^2	RMSE
Uncorrected	0.031	0.984	-0.007	1.004
IPW	0.038	0.981	-0.005	1.002
Dragonnet	-	-	-	-
CA (ours)	**0.224**	**0.881**	**0.200**	**0.894**
Improvement	18.5%	10.0%	20.5%	10.8%

Table 16: Results on regression and multi-class classification tasks with the language modality only (unbiased evaluation dataset). Our RMSE metric is normalized RMSE, or RMSE divided by the standard deviation of the evaluation set.