CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations

Hangzhi Guo
The Pennsylvania State University
University Park, PA, USA
hangz@psu.edu

Thanh H. Nguyen
University of Oregon
Eugene, OR, USA
thanhhng@cs.uoregon.edu

Amulya Yadav
The Pennsylvania State University
University Park, PA, USA
amulya@psu.edu

ABSTRACT

This work presents *CounterNet*, a novel *end-to-end* learning framework which integrates Machine Learning (ML) model training and the generation of corresponding counterfactual (CF) explanations into a single end-to-end pipeline. Counterfactual explanations offer a contrastive case, i.e., they attempt to find the smallest modification to the feature values of an instance that changes the prediction of the ML model on that instance to a predefined output. Prior techniques for generating CF explanations suffer from two major limitations: (i) all of them are *post-hoc methods* designed for use with proprietary ML models — as a result, their procedure for generating CF explanations is uninformed by the training of the ML model, which leads to misalignment between model predictions and explanations; and (ii) most of them rely on solving separate time-intensive optimization problems to find CF explanations for each input data point (which negatively impacts their runtime). This work makes a novel departure from the prevalent post-hoc paradigm (of generating CF explanations) by presenting *CounterNet*, an *end-to-end* learning framework which integrates predictive model training and the generation of counterfactual (CF) explanations into a single pipeline. Unlike post-hoc methods, CounterNet enables the optimization of the CF explanation generation only *once* together with the predictive model. We adopt a block-wise coordinate descent procedure which helps in effectively training CounterNet's network. Our extensive experiments on multiple real-world datasets show that CounterNet generates high-quality predictions, and consistently achieves 100% CF validity and low proximity scores (thereby achieving a well-balanced cost-invalidity trade-off) for any new input instance, and runs 3X faster than existing state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*.

KEYWORDS

Counterfactual Explanation, Algorithmic Recourse, Explainable Artificial Intelligence, Interpretability

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599290

ACM Reference Format:
Hangzhi Guo, Thanh H. Nguyen, and Amulya Yadav. 2023. CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3580305.3599290

1 INTRODUCTION

Most prior work in Explainable Artificial Intelligence (XAI) has been focused on developing techniques to interpret decisions made by black-box machine learning (ML) models. For example, widely known approaches rely on attribution-based explanations for interpreting an ML model (e.g., LIME [35] and SHAP [24]). These approaches can help computer scientists and ML experts understand why (and how) ML models make certain predictions. However, end users (who generally have no ML expertise) are often more interested in understanding actionable implications of the ML model's predictions (as it relates to them), rather than just understanding how these models arrive at their predictions. For example, if a person applies for a loan and gets rejected by a bank's ML algorithm, he/she might be more interested in knowing what they need to change in a future loan application in order to successfully get a loan, rather than understanding how the bank's ML algorithm makes all of its decisions.

Thus, from an end-user perspective, counterfactual (CF) explanation techniques [1] [53] may be more preferable. A CF explanation offers a contrastive case — to explain the predictions made by an ML model on data point x, CF explanation methods find a new *counterfactual* point (or example) x', which is close to x but gets a different (or opposite) prediction from the ML model. CF explanations (or CF examples) [2] are useful because they can be used to offer recourse to vulnerable groups. For example, when an ML model spots a student as being vulnerable to dropping out from school, CF explanation techniques can suggest corrective measures to teachers, who can intervene accordingly.

Generating high-quality CF explanations is a challenging problem because of the need to balance the *cost-invalidity trade-off* [34] between: (i) the *invalidity*, i.e., the probability that a CF example is invalid, or it does not achieve the desired (or opposite) prediction from the ML model; and (ii) the *cost of change*, i.e., the amount of modifications required to convert input instance x into CF example x' (as measured by the distance between x and x'). Figure 1

[1] Counterfactual explanations are closely related to algorithmic recourse [48] and contrastive explanations [8]. Although these terms are proposed under different contexts, their differences to CF explanations have been blurred [45, 50], i.e. these terms are used interchangeably. Further, the literature on counterfactual explanations is not directly linked to "counterfactuals" in causal inference.
[2] We use *CF explanations* and *CF examples* interchangeably in the rest of this paper.

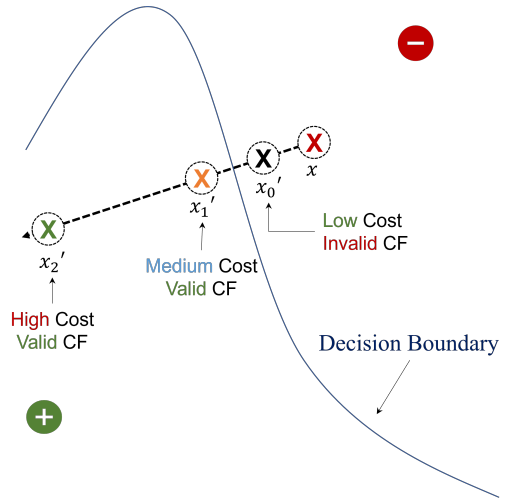


Figure 1: Illustration of the cost-invalidity trade-off in CF explanations for binary classification problems. This illustrates that balancing the *cost-invalidity* trade-off is key to generating high-quality CF explanations. For example, both x_0' and x_2' do not properly balance this trade-off (i.e., it is invalid/valid but has low/high cost, respectively). On the other hand, x_1' is a near-optimal CF explanation that balances this trade-off (i.e., it is valid with medium cost). To optimally balance this trade-off, it is important to have access to the decision boundary of the ML model.

illustrates this trade-off by showing three different CF examples for an input instance x. If *invalidity* is ignored (and optimized only for *cost of change*), the generated CF example can be trivially set to x itself. Conversely, if *cost of change* is ignored (and optimized only for *invalidity*), the generated CF example can be set to x_2' (or any sufficiently distanced instance with different labels). More generally, CF examples with high (low) invalidities usually imply low (high) *cost of change*. To optimally balance this trade-off, it is critical for CF explanation methods to have access to the decision boundary of the ML model, without which finding a near-optimal CF explanation (i.e., x_1') is difficult. For example, it is difficult to distinguish between x_1' (a valid CF example) and x_0' (an invalid CF example) without prior knowledge of the decision boundary.

Existing CF explanation methods suffer from three major limitations. First, to our best knowledge, all prior methods belong to the *post-hoc* explanation paradigm, i.e., they assume a trained black-box ML model as input. This post-hoc assumption has certain advantages, e.g., post-hoc explanation techniques are often agnostic to the particulars of the ML model, and hence, they are generalizable enough to interpret any *third-party* proprietary ML model. However, we argue that in many real-world scenarios, the model-agnostic approach provided by post-hoc CF explanation methods is not desirable. With the advent of data regulations that enshrine the "*Right to Explanation*" (e.g., EU-GDPR [53]), service providers are required by law to communicate both the decision outcome (i.e., the ML model's prediction) and its actionable implications (i.e., a CF explanation for this prediction) to an end-user. In these scenarios, the post-hoc assumption is overly limiting, as service providers can build specialized CF explanation techniques that can leverage the knowledge of their particular ML model to generate higher-quality CF explanations. Second, in the post-hoc CF explanation paradigm, the optimization procedure that finds CF explanations is completely uninformed by the ML model training procedure (and the resulting decision boundary). Consequently, such a post-hoc procedure does not properly balance the cost-invalidity trade-off (as explained above), causing shortcomings in the quality of the generated CF explanations (as shown in Section 4). Finally, most CF explanation methods are very slow — they search for CF examples by solving a separate time-intensive optimization problem for each input instance [18, 29, 53], which is not viable in time-constrained environments, e.g., runtime is a critical factor when such techniques are deployed to end-user facing devices such as smartphones [2, 58].

Contributions. We make a novel departure from the prevalent post-hoc paradigm of generating CF explanations by proposing *CounterNet*, a learning framework that combines the training of the ML model and the generation of corresponding CF explanations into a single end-to-end pipeline (i.e., from input to prediction to explanation). CounterNet has three contributions:

- Unlike post-hoc approaches (where CF explanations are generated after the ML model is trained), CounterNet uses a (neural network) model-based CF generation method, enabling the joint training of its CF generator network and its predictor network. At a high level, CounterNet's CF generator network takes as input the learned representations from its predictor network, which is jointly trained along with the CF generator. This joint training is key to achieving a well-balanced cost-invalidity trade-off (as we show in Section 4).
- We theoretically analyze CounterNet's objective function to show two key challenges in training CounterNet: (i) poor convergence of learning; and (ii) a lack of robustness against adversarial examples. To remedy these issues, we propose a novel block-wise coordinate descent procedure.
- We conduct extensive experiments which show that CounterNet generates CF explanations with ~100% validity and low cost of change (~9.8% improvement to baselines), which shows that CounterNet balances the cost-invalidity trade-off significantly better than baseline approaches. In addition, this joint-training procedure does not sacrifice CounterNet's predictive accuracy and robustness. Finally, CounterNet runs orders of magnitude (~3X) faster than baselines.

2 RELATED WORK

Broadly speaking, to ensure that models' predictions are interpretable to end-users, two distinct approaches have been proposed in prior work: (i) applying "glass-box" ML models (e.g., decision trees, rule lists, etc.) that are intrinsically interpretable [6, 22, 23, 38]; and (ii) applying "black-box" ML models, and explaining their predictions in a post-hoc manner [7, 35, 53]. Here, we focus our discussion on black-box model approaches, as the interpretability of "glass-box" models often comes at the cost of decreased predictive accuracy [1], which limits the real-world usability of these methods.

2.1 Explaining Black-Box Models

Attribution Based Explanation. There exists a lot of prior work on explanation techniques for black-box ML models. One primary approach is to explain the predictions made by an ML model by highlighting the importance of attributions for each data instance. For example, Ribeiro et al. [35] introduced LIME, which generates local explanations by sampling data near the input instance, and then uses a linear model to fit this data (which is then used to generate the explanation via attribution). Similarly, Lundberg and Lee [24] introduced SHAP, a unified explanation framework to find locally faithful explanations by using the Shapley value concept in game theory. Furthermore, for interpreting predictions made by deep neural networks, gradient-based saliency maps are often adopted to understand attribution importances [41, 43, 46].

Case-Based Explanations. Another field in ML model interpretation is on *case-based explanations* which conveys model explanations by providing (similar) data samples to the human end-user [11, 28, 30]. For example, Chen et al. [7] propose a novel explanation style, "*this looks like that*", to explain image classifications by identifying similar images (and their regions) in the dataset. Koh and Liang [19] adopt influence functions to identify influential data points in the training set that are used for generating predictions on test instances. However, both attribution- and case-based methods are of limited utility to average end-users, who are often more interested in understanding actionable implications of these model predictions (as it relates to them), rather than understanding decision rules used by ML models for generating predictions.

Counterfactual Explanations. Our work is most closely related to prior literature on counterfactual explanation techniques, which focuses on generating/finding new instances that lead to different predicted outcomes [17, 45, 50, 53]. Counterfactual explanations are preferred by human end-users as these explanations provide actionable recourse in many domains [4, 5, 27]. Almost all prior work in this area belongs to the post-hoc CF explanation paradigm, which we categorize into *non-parametric* and *parametric* methods:

- **Non-parametric methods.** Non-parametric methods aim to find a counterfactual explanation without the use of parameterized models. Wachter et al. [53] proposed *VanillaCF* which generates CF explanations by minimizing the distance between the input instance and the CF example, while pushing the new prediction towards the desired class. Other algorithms, built on top of *VanillaCF*, optimize other aspects, such as recourse cost [48], fairness [52], diversity [29], closeness to the data manifold [49], causal constraints [18], uncertainty [39], and robustness to model shift [47]. However, this line of work is inherently post-hoc and relies on solving a separate optimization problem for each input instance. Consequently, running them is time-consuming, and their post-hoc nature leads to poor balancing of the cost-invalidity trade-off.
- **Parametric methods.** These methods use parametric models (e.g., a neural network model) to generate CF explanations. For example, Joshi et al. [16], Pawelczyk et al. [32] generate CF explanations by perturbing the latent variable of a variational autoencoder (VAE) model. Similarly, Nemirovsky et al. [31], Singla et al. [42], Yang et al. [56], Guyomard et al. [12], Mahajan et al. [26], and Rodríguez et al. [36] train generative models (GAN and VAE, respectively) to produce CF explanations for a trained ML model. However, these methods are still post-hoc in nature, and thus, they also suffer from poorly balanced cost-invalidity trade-offs. Contrastingly, we depart from this post-hoc paradigm, which leads to a greater alignment between CounterNet's predictions and CF explanations. Note that Ross et al. [37] proposed a recourse-friendly ML model by integrating recourse training during predictive model training. However, their work does not focus on generating CF explanations. In contrast, we focus on generating predictions and CF explanations simultaneously.

3 THE PROPOSED FRAMEWORK: COUNTERNET

Unlike prior work, our proposed framework CounterNet relies on a novel integrated architecture which combines predictive model training and counterfactual explanation generation into a single optimization framework. Through this integration, we can simultaneously optimize the accuracy of the trained predictive model and the quality of the generated counterfactual explanations.

Formally, given an input instance $x \in \mathbb{R}^d$, CounterNet aims to generate two outputs: (i) the ML prediction component outputs a prediction $\hat{y}_x$ for input instance x; and (ii) the CF explanation generation component produces a CF example $x' \in \mathbb{R}^d$ as an explanation for input instance x. Ideally, the CF example x' should get a different (and often more preferable) prediction $\hat{y}_{x'}$, as compared to the prediction $\hat{y}_x$ on the original input instance x (i.e., $\hat{y}_{x'} \neq \hat{y}_x$). In particular, if the desired prediction output is binary-valued $(0, 1)$, then $\hat{y}_x$ and $\hat{y}_{x'}$ should take on opposite values (i.e., $\hat{y}_x + \hat{y}_{x'} = 1$).

3.1 Network Architecture

Figure 2 illustrates CounterNet's architecture which includes three components: (i) an encoder network $h(\cdot)$; (ii) a predictor network $f(\cdot)$; and (iii) a CF generator network $g(\cdot)$. During training, each input instance $x \in \mathbb{R}^d$ is first passed through the encoder network to generate a dense latent vector representation of x (denoted by $z_x = h(x)$). Then, this latent representation is passed through both the predictor network and the CF generator network. The predictor network outputs a softmax representation of the prediction $\hat{y}_x = f(z_x)$. To generate CF examples, the CF generator network takes two pieces of information: (i) the final representation of the predictor network p_x (before it is passed through the softmax layer), and (ii) the latent vector z_x (which contains a dense representation of the input x). These two vectors are concatenated to produce the final latent vector $z'_x = p_x \oplus z_x$, which is passed through the CF generator network to produce a CF example $x' = g(z'_x)$. Note that the final learned representation of the predictor network p_x (for input x) reveals useful information about the decision boundary that is being learned by the predictor network $f(\cdot)$. Therefore, passing the final representation p_x as an input into the CF generator network implicitly conveys some kind of information about the decision boundary (that is being learned by the predictor network) to the CF generation procedure, and this information is leveraged by the CF generator network to find high-quality CF examples x'

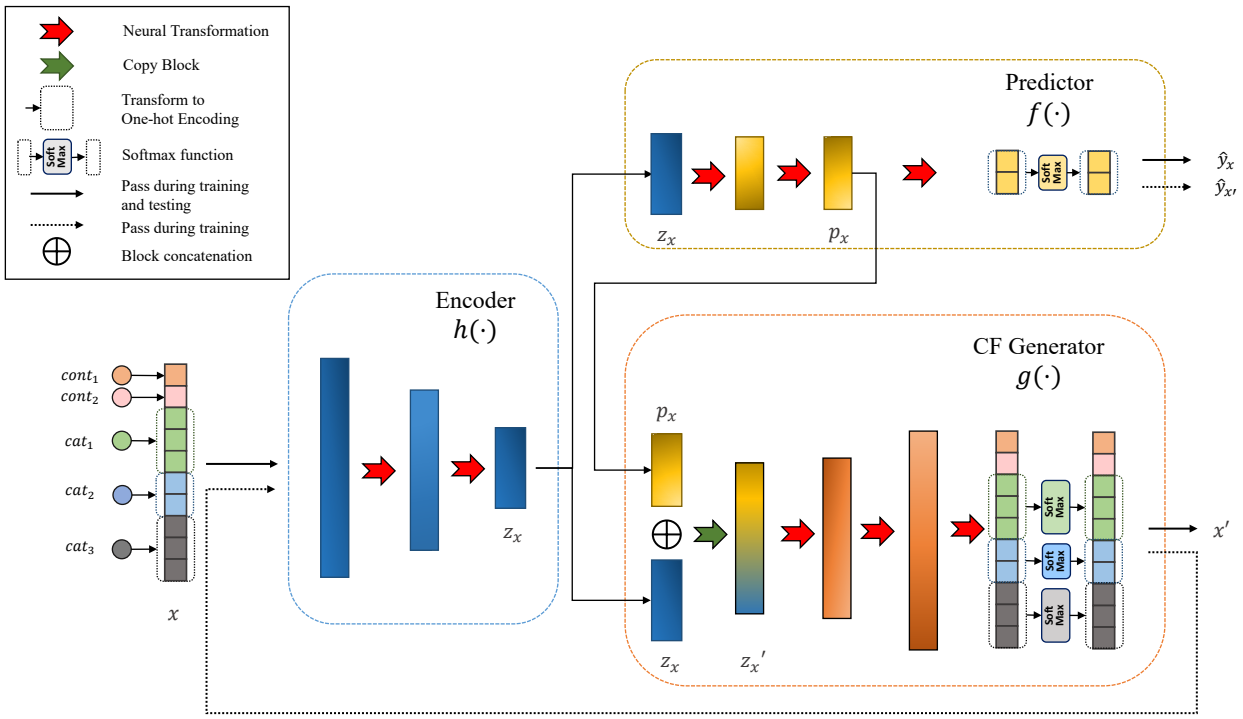


Figure 2: CounterNet contains three components: an encoder to transform the input into a dense latent vector, a predictor network to output the prediction, and a CF generator to produce explanations.

that achieve a better balance on the cost-invalidity trade-off (we validate this in Section 4).

Furthermore, to ensure that the CF generator network outputs *valid* CF examples (i.e., $\hat{y}_x \neq \hat{y}_{x'}$), the output of the CF generator network x' is also passed back as an input through the encoder and predictor networks when training CounterNet. This additional feedback loop (from the output of CF generator network back into the encoder and predictor networks) is necessary to optimize the *validity* of generated CF examples (intuitively speaking, in order to ensure that the predictions for input x and CF example x' are opposite, the CF example x' generated by the CF generator network needs to be passed back through the predictor network). As such, we can now train the entire network in a way such that the predictor network outputs opposite predictions $\hat{y}_x$ and $\hat{y}_{x'}$ for the input instance x and the CF example x', respectively. Note that this "*feedback loop*" connection is only needed during training, and is removed at test time. This design aims to achieve a better balance on the cost-invalidity tradeoff (as shown in Section 4).

Design of Encoder, Predictor & CF Generator. All three components in CounterNet's architecture consist of a multi-layer perception (MLP) [3]. The encoder network in CounterNet consists of two feed-forward layers that down-sample to generate a latent vector $z \in \mathbb{R}^k$ (s.t. $k < d$). The predictor network passes this latent vector z through two feed-forward layers to produce the predictor representation p. Finally, the predictor network outputs the probability distribution over predictions with a fully-connected layer followed by a softmax layer. On the other hand, the CF generator network takes the final latent representation $z' = z \oplus p$ as an input, and up-samples to produce CF examples $x' \in \mathbb{R}^d$.

Each feed-forward neural network layer inside CounterNet uses LeakyRelu activation functions [55] followed by a dropout layer [44] to avoid overfitting. Note that the number of feed-forward layers, the choice of activation function, etc., were hyperparameters that were optimized using grid search (See Appendix B.2).

Handling Categorical Features. To handle categorical features, we customize CounterNet's architecture for each dataset. First, we transform all categorical features in each dataset into numeric features via one-hot encoding. In addition, for each categorical feature, we add a softmax layer after the final output layer in the CF generator network (Figure 2), which ensures that the generated CF examples respect the one-hot encoding format (as the output of the softmax layer will sum up to 1). Finally, we normalize all continuous features to the $[0, 1]$ range before training.

3.2 CounterNet Objective Function

We now describe the three-part loss function that is used to train the network architecture outlined in Figure 2. Each part of our loss function corresponds to a desirable objective in CounterNet's output: (i) *predictive accuracy* - the predictor network should output

[3]CounterNet can work with alternate neuronal blocks, e.g., convolution, attention, although achieving state-of-the-art training and performance of these neuronal blocks is a topic for future work (see Appendix E for details).

accurate predictions $\hat{y}_x$; (ii) *counterfactual validity* - CF examples x' produced by the CF generator network should be valid, i.e., they get opposite predictions from the predictor network (e.g. $\hat{y}_x + \hat{y}_{x'} = 1$); and (iii) *minimizing cost of change* - minimal modifications should be required to change input instance x to CF example x'. Thus, we formulate this multi-objective minimization problem to optimize the parameter of overall network θ:

$$\begin{aligned}\mathcal{L}_1 &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_{x_i})^2 \\ \mathcal{L}_2 &= \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_{x_i} - (1 - \hat{y}_{x'_i}))^2 \\ \mathcal{L}_3 &= \frac{1}{N}\sum_{i=1}^{N}(x_i - x'_i)^2\end{aligned} \tag{1}$$

where N denotes the number of instances in our dataset, the prediction loss $\mathcal{L}_1$ denotes the mean squared error (MSE) between the actual and the predicted labels (y_i and $\hat{y}_{x_i}$ on instance x_i, respectively), which aims to maximize predictive accuracy. Similarly, the validity loss $\mathcal{L}_2$ denotes the MSE between the prediction on instance x_i (i.e., $\hat{y}_{x_i}$), and the opposite of the prediction received by the corresponding CF example x'_i (i.e., $1 - \hat{y}_{x'_i}$). Intuitively, minimizing $\mathcal{L}_2$ maximizes the validity of the generated CF example x'_i by ensuring that the predictions on x'_i and x_i are different. Finally, the proximity loss $\mathcal{L}_3$ represents the MSE distance between input instance x_i and the CF example x'_i, which aims to minimize proximity (or cost of change).

Note that we choose MSE loss (instead of the conventional choice of using cross entropy) for $\mathcal{L}_1$ and $\mathcal{L}_2$ because our experimental analysis suggests that this choice of loss functions is crucial to CounterNet's superior performance, as replacing $\mathcal{L}_1$ and $\mathcal{L}_2$ with binary cross-entropy functions leads to degraded performance (as we show in ablation experiments in Section 4). In fact, our choice of MSE based loss functions is supported by similar findings in prior research, which shows that MSE loss-based training seems to be less sensitive to randomness in initialization [15], more robust to noise [10], and less prone to overfitting [3] on a wide variety of learning tasks (as compared to cross-entropy loss).

Given these three loss components, we aim to optimize the parameter θ of the overall network, which can be formulated as the following minimization problem:

$$\underset{\theta}{\text{argmin}}\ \ \lambda_1 \cdot \mathcal{L}_1 + \lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3 \tag{2}$$

where (λ_1, λ_2, λ_3) are hyper-parameters to balance the three loss components. Unfortunately, directly solving Eq. 2 as-is (via gradient descent) leads to poor convergence and degraded adversarial robustness. In the next section, we theoretically analyze the cause of these challenges, and propose a blockwise coordinate descent procedure to remedy these issues.

3.3 Training Procedure

The conventional way of solving the optimization problem in Eq. 2 is to use gradient descent with backpropagation (BP). However, directly optimizing the objective function (Eq. 2) results in two fundamental issues: (1) *poor convergence in training* (shown in Lemma 3.1), and (2) *proneness to adversarial examples* (shown in Lemma 3.2).

Issue I: Poor Convergence. Optimizing Eq. 2 as-is via BP leads to poor convergence. This occurs because Eq. 2 contains two different loss objectives with divergent gradients, as Lemma 3.1 shows the gradients of $\mathcal{L}_1$ and $\mathcal{L}_2$ move in opposite directions. Consequently, the accumulated gradient direction (i.e., gradient across all three loss objectives) fluctuates drastically, which leads to poor convergence of training (as we show in Table 5 in Section 4).

Lemma 3.1 (Divergent Gradient Problem). *Let $\mathcal{L}_1 = \|y - \hat{y}_x\|_2$, and $\mathcal{L}_2 = \|\hat{y}_x - (1 - \hat{y}_{x'})\|_2$, assuming that $x' \to x$, $0 < \hat{y}_x < 1$, y is a binary label, and $|\hat{y}_x - y| < 0.5$, then $\nabla\mathcal{L}_1 \cdot \nabla\mathcal{L}_2 < 0$. (See proof in Appendix A.1)*

Issue II: Adversarial Examples. Our training procedure should generate high-quality CF examples x' for input instances x without sacrificing the adversarial robustness of the predictor network. Unfortunately, optimizing Eq. 2 as-is is at odds with the goal of achieving adversarial robustness (we show it empirically in Figure 4). Lemma 3.2 diagnoses the cause of poor adversarial robustness - it shows that optimizing $\mathcal{L}_2$ with respect to the predictive weights θ_f decreases the robustness of the predictor $f(\cdot)$ (by increasing the Lipschitz constant of $f(\cdot)$, which leads to an increased vulnerability to adversarial examples as found in prior research [14, 40, 54]).

Lemma 3.2 (Lipschitz Continuity). *Suppose f is a locally Lipschitz continuous function parameterized by θ_f, then it satisfies $|f_{\theta_f}(x) - f_{\theta_f}(x')| \leq K\|x - x'\|_2$, where the Lipschitz constant of f is $K = \sup_{x' \in \mathbb{B}(x,\epsilon)}\{\|\nabla f_{\theta_f}(x')\|_2\}$. Let $\mathcal{L}_2 = \left\|f_{\theta_f}(x) - (1 - f_{\theta_f}(x'))\right\|_2$, assuming that $x' \to x$, $0 < f_{\theta_f}(\cdot) < 1$, $f_{\theta_f}(x) \to y$, and y is a binary label, then minimizing $\mathcal{L}_2$ w.r.t. θ_f increases the Lipschitz constant K. (See proof in Appendix A.2)*

Training Procedure. We propose a block-wise coordinate descent procedure to remedy these two issues. This block-wise coordinate descent procedure divides the problem of optimizing Eq. 2 into two parts: (i) optimizing predictive accuracy (primarily influenced by $\mathcal{L}_1$); and (ii) optimizing the validity and proximity of CF generation (primarily influenced by $\mathcal{L}_2$ and $\mathcal{L}_3$). Specifically, for each mini-batch of m data points $\{x^{(i)}, y^{(i)}\}^m$, we apply two gradient updates to the network through backpropagation. For the first update, we compute $\theta^{(1)} = \theta^{(0)} - \nabla_{\theta^{(0)}}(\lambda_1 \cdot \mathcal{L}_1)$, and for the second update, we compute $\theta_g^{(2)} = \theta_g^{(1)} - \nabla_{\theta_g^{(1)}}(\lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3)$.

This block-wise coordinate descent procedure mitigates the aforementioned issues as follows: (1) it *handles poor convergence in training* by ensuring that the gradient of $\mathcal{L}_1$ and $\mathcal{L}_2$ are calculated separately. Because $\nabla\mathcal{L}_1$ and $\nabla\mathcal{L}_2$ are back-propagated to the network at different stages, CounterNet does not suffer from the divergent gradient problem (Lemma 3.1), and this procedure leads to significantly better convergence of training. (2) Moreover, it *improves adversarial robustness* of our predictor network. During the second stage of our coordinate descent procedure (when we optimize for $(\lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3)$), we only update the weights in the CF generator θ_g and freeze gradient updates in both the encoder θ_h and predictor θ_f networks. As shown in Lemma 3.2, when optimizing the predictor and CF generator simultaneously, it will unwantedly increase the Lipschitz constant of the predictor network. By separately updating the gradient of the encoder, predictor, and CF generator, it ensures that the Lipschitz constant of

Table 1: Summary of Datasets used for Evaluation.

Dataset	Size	#Continuous	#Categorical
Adult	32,561	2	6
Credit	30,000	20	3
HELOC	10,459	21	2
OULAD	32,593	23	8

the predictor network does not increase, which in turn, improves the adversarial robustness of the predictor network.

4 EXPERIMENTAL EVALUATION

We primarily focus our evaluation on heterogeneous tabular datasets for binary classification problems (which is the most common and reasonable setting for CF explanations [45, 50]). However, CounterNet can be applied to multi-class classification settings, and it can also be adapted to work with other modalities of data.

Baselines. We compare CounterNet against eight state-of-the-art CF explanation methods:

- *VanillaCF* [53] is a non-parametric post-hoc method which generates CF examples by optimizing CF validity and proximity;
- *DiverseCF* [29], *ProtoCF* [49], and *UncertainCF* [39] are non-parametric methods which optimize for diversity, consistency with prototypes, and uncertainty, respectively;
- *VAE-CF* [26], *CounteRGAN* [31], *C-CHVAE* [32], and *VCNet* [12] are parametric methods which use generative models (i.e., VAE or GAN) to generate CF examples [4].

Unlike CounterNet, all of the post-hoc methods require a trained predictive model as input. Thus, for each dataset, we train a neural network model and use it as the target predictive model for all baselines. For a fair comparison, we only keep the encoder and predictor network inside CounterNet's architecture (Figure 2), and optimize them for predictive accuracy alone (i.e., $\mathcal{L}_1$). This combination of encoder and predictor networks is then used as the black-box predictive model for our baselines.

Datasets. To remain consistent with prior work on CF explanations [50], we evaluate CounterNet on four benchmarked real-world binary classification datasets. Table 1 summarizes these four datasets.

- *Adult* [20] which aims to predict whether an individual's income reaches \$50K (`Y=1`) or not (`Y=0`) using demographic data;
- *Credit* [57] which uses historical payments to predict the default of payment (`Y=1`) or not (`Y=0`);
- *HELOC* [9] which predicts if a homeowner qualifies for a line of credit (`Y=1`) or not (`Y=0`);
- *OULAD* [21]: which predicts whether MOOC students drop out (`Y=1`) or not (`Y=0`), based on their online learning logs.

Evaluation Metrics. For each input x, CF explanation methods generate two outputs: (i) a prediction $\hat{y}_x$; and (ii) a CF example x'. We evaluate the quality of both these outputs using separate metrics. For evaluating predictions, we use *predictive accuracy* (as all four datasets are fairly class-balanced). High *predictive accuracy* is desirable as we do not want to provide explanations for incorrect predictions from ML models.

[4]Note that Yang et al. [56] propose another parametric post-hoc method, but we exclude it in our baseline comparison because it achieves comparable performance to C-CHVAE on benchmarked datasets (as reported in [56]).

Table 2: Predictive accuracy of CounterNet. CounterNet achieves comparable predictive performance as base models.

Dataset	Base Model	CounterNet
Adult	0.831	0.828
Credit	0.813	0.819
HELOC	0.717	0.716
OULAD	0.934	0.929

For evaluating CF examples, we use five widely used metrics from prior literature:

- *Validity* is defined as the fraction of input instances on which CF explanation techniques output valid counterfactual examples, i.e., the fraction of input data points for which $\hat{y}_x + \hat{y}_{x'} = 1$. High *validity* is desirable, as it implies the technique's effectiveness at creating valid CF examples. This is a widely-used metric in prior CF explanation literature [26, 29, 47].
- *Proximity* is defined as the L_1 norm distance between x and x' divided by the number of features. The *proximity* metric measures the quality of CF examples as it is desirable to have fewer modifications in the input space to convert it into a valid CF example [26, 29, 53].
- *Sparsity* measures the number of feature changes (i.e., L_0 norm) between x and x'. This metric stems from the motivation of *sparse explanations*, i.e., CF explanations are more interpretable to end-users if they require changes to fewer features [27, 33, 53]. Both *proximity* and *sparsity* serve as proxies for measuring the *cost of change* of our CF explanation approach, as it is desirable to have fewer modifications in the input space to convert it into a valid CF example.
- *Manifold distance* is the L_1 distance to the k-nearest neighbor of x' (we use $k = 1$ to remain consistent with prior work [51]). Low *manifold distance* is desirable as closeness to the training data manifold indicates realistic CF explanations [49, 51].
- Finally, we also report the *runtime* for generating CF examples.

4.1 Evaluation of CounterNet Performance

Predictive Accuracy. Table 2 compares CounterNet's predictive accuracy against the base prediction model used by baselines. This table shows that CounterNet exhibits highly competitive predictive performance - it achieves marginally better accuracy on the Credit dataset (row 2), and achieves marginally lower accuracy on the remaining datasets. Across all four datasets, the difference between the predictive accuracy of CounterNet and the base model is $\sim 0.1\%$. Thus, the potential benefits achieved by CounterNet's joint training of predictor and CF generator networks do not come at a cost of reduced predictive accuracy.

Counterfactual Validity. Table 3 compares the validity achieved by CounterNet and baselines on all four datasets. We observe that CounterNet, C-CHVAE, and VCNet are the only three methods with 100% validity on all datasets. With respect to the other baselines,

Table 3: Evaluation of CF explanations: CounterNet achieves perfect validity (i.e., Val.), and it incurs comparable (or lesser) cost of changes (i.e., Prox, Spar.) than baseline methods, with comparable manifold distance (i.e., Man.). Bold and *italicized* cells highlight the best and second-best performing methods, respectively.

Method	Adult				Credit				HELOC				OULAD			
	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.
VanillaCF	0.76	.202	**.556**	*0.57*	0.92	**.123**	.841	0.59	**1.00**	.154	.883	0.71	**1.00**	.101	.762	1.30
DiverseCF	0.54	.276	.662	1.16	**1.00**	.264	.918	1.68	0.90	.149	*.434*	1.34	0.68	.117	**.565**	2.51
ProtoCF	0.59	.250	.648	0.62	0.92	.197	.855	0.82	**1.00**	.168	.805	*0.56*	**1.00**	.107	.754	1.46
UncertainCF	0.36	.307	.713	1.23	0.62	.155	**.217**	0.80	0.55	.130	**.161**	0.94	0.59	.098	.734	2.23
C-CHVAE	**1.00**	.281	.721	0.94	**1.00**	.357	.853	1.85	**1.00**	.155	.790	0.81	**1.00**	.110	.797	2.11
VAE-CF	0.66	.287	.734	1.03	0.13	.201	.756	0.62	**1.00**	.221	.893	1.04	**1.00**	.115	.586	2.19
CounteRGAN	0.78	.327	.698	2.21	0.39	.260	.687	2.03	**1.00**	.271	.509	2.23	0.43	.087	.587	2.15
VCNet	**1.00**	.291	.755	**0.19**	**1.00**	.162	.939	**0.16**	**1.00**	.154	.786	**0.39**	**1.00**	.095	.903	1.33
CounterNet	**1.00**	**.196**	*.644*	0.64	**1.00**	*.132*	.912	*0.56*	**1.00**	**.125**	.740	*0.56*	**1.00**	**.075**	.725	**0.87**

CounterNet achieves 8% and 12.3% higher average validity (across all datasets) than VanillaCF and ProtoCF (our next best baselines).

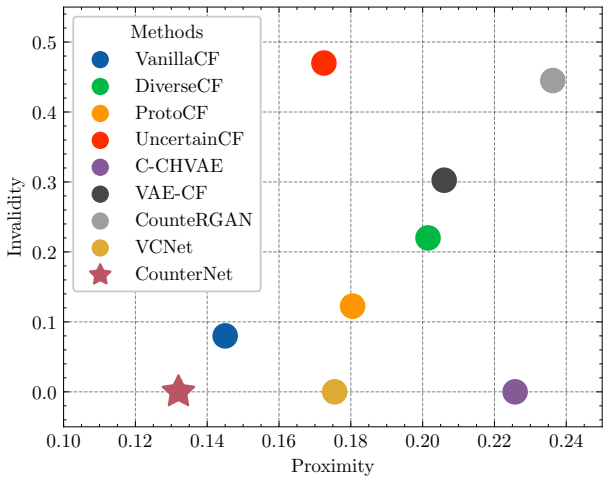


Figure 3: Illustration of the cost-invalidity trade-off across all four datasets. Methods at the bottom left are preferable. On average, CounterNet achieves the lowest invalidity and proximity (i.e., cost) across all four datasets.

Proximity & Sparsity. Table 3 compares the proximity/sparsity achieved by all CF explanation methods. CounterNet achieves at least 3% better proximity than all other baselines on three out of four datasets (Adult, HELOC, and OULAD), and it is the second best performing model on the Credit dataset (where it achieves 7.3% poorer proximity than VanillaCF). In terms of sparsity, CounterNet performs reasonably well; it is the second-best-performing model on the Adult and HELOC datasets even though CounterNet does not explicitly optimize for sparsity. *This shows that CounterNet outperforms all baselines by generating CF examples with the highest validity and best proximity scores.*

Cost-Invalidity Trade-off. We illustrate the cost-invalidity trade-off [34] for all methods. Figure 3 shows this trade-off by plotting their average proximity/invalidity values. This figure shows that CounterNet lies on the bottom left of this figure — it consistently achieves the lowest invalidity and cost on all four datasets. In comparison, VCNet achieves the same perfect validity, but at the expense of ~34% higher cost than CounterNet. Similarly, C-CHVAE demands ~71% higher cost than CounterNet to achieve perfect validity. On the other hand, VanillaCF achieves a comparable cost to CounterNet (10% higher cost), but it achieves lower validity by 8%. This result highlights that CounterNet's joint training enables it to properly balance the cost-invalidity trade-off.

Manifold Distance. Table 3 shows that CounterNet achieves the second-lowest manifold distance on average (right below VCNet, which explicitly optimizes for data manifold). In particular, CounterNet achieves the lowest manifold distance in *OULAD*, and is ranked second in *Credit* and *HELOC*. This result shows that CounterNet generates highly realistic CF examples that adhere to the data manifold, despite not optimizing for the manifold distance.

Running Time. Table 4 shows the average runtime (in milliseconds) of different methods to generate a CF example for a single data point. CounterNet outperforms all eight baselines in every dataset. In particular, CounterNet generates CF examples ~3X faster than VAE-CF, CouneRGAN, and VCNet, ~5X faster than C-CHVAE, and three orders of magnitude (>1000X) faster than other baselines. This result shows that CounterNet is more usable for adoption in time-constrained environments.

4.2 Further Analysis

Ablation Analysis. We analyze five ablations of CounterNet to underscore the design choices inside CounterNet. First, we accentuate the importance of the MSE loss functions used to optimize CounterNet (Eq. 2) by replacing the MSE based $\mathcal{L}_1$ and $\mathcal{L}_2$ loss in

Table 4: Runtime comparison (in milliseconds). CounterNet runs *faster* than all of the baselines in all four datasets.

Method	Adult	Credit	HELOC	OULAD
VanillaCF	1432.09	1358.26	1340.42	1705.93
DiverseCF	4685.39	3898.43	3921.72	5478.17
ProtoCF	2348.21	2056.01	1956.71	2823.29
UncertainCF	379.95	60.80	7.91	6.81
C-CHVAE	3.28	568.28	2.68	4.79
VAE-CF	1.72	1.28	1.48	1.84
CounteRGAN	1.96	1.77	1.59	2.40
VCNet	1.39	1.23	1.13	1.81
CounterNet	**0.64**	**0.39**	**0.44**	**0.79**

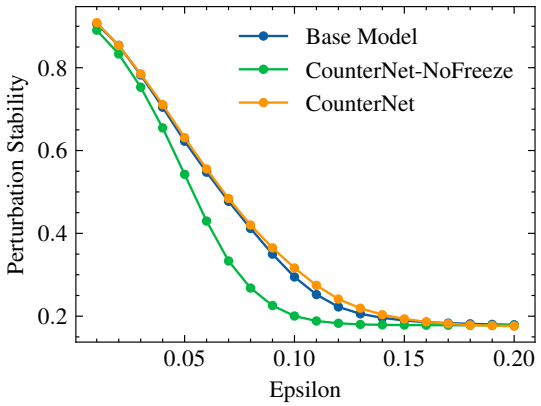


Figure 4: Adversarial robustness of the predictor $f(\cdot)$ under the PGD attack [25] on the adult dataset (higher is better). CounterNet is comparable to the base model in robustness. In comparison, *CounterNet-NoFreeze* achieves poorer robustness than CounterNet. This shows that CounterNet does not lead to increased vulnerability to adversarial examples.

Eq. 2 with binary cross entropy loss (*CounterNet-BCE*). Second, we underscore the importance of CounterNet's two-stage coordinate descent procedure by using conventional one-step BP optimization to train CounterNet instead (*CounterNet-SingleBP*). In addition, we validate CounterNet's architecture design by experimenting two alternative designs: (i) we use a *separate* predictor $f : \mathcal{X} \rightarrow \mathcal{Y}$ and CF generator $g : \mathcal{X} \rightarrow \mathcal{X}'$, such that f and g share no identical components (unlike in CounterNet, where z_x are shared with both f and g; *CounterNet-Separate*); and (ii) we highlight the design choice of passing p_x to the CF generator by excluding passing p_x (*CounterNet-NoPass-p_x*). Finally, we highlight the importance of the joint-training of predictor and CF generator in CounterNet by training the CounterNet in a post-hoc fashion (*CounterNet-Posthoc*), i.e., we first train the predictor on the *entire* training dataset, and optimize CF generator while the trained predictor is frozen.

Table 5 compares the validity and proximity achieved by CounterNet and five ablations. Importantly, each ablation leads to degraded performance as compared to CounterNet, which demonstrates the importance of CounterNet's different design choices. *CounterNet-BCE* and *CounterNet-SingleBP* perform poorly in comparison, which illustrates the importance of the MSE-based loss function and block-wise coordinate descent procedure. Similarly, *CounterNet-Separate* and *CounterNet-NoPass-p_x* achieve degraded validity and proximity, which highlight the importance of CounterNet's architecture design. Finally, *CounterNet-Posthoc* achieves comparable validity as CounterNet, but fails to match the performance of proximity. This result demonstrates the importance of the joint-training procedure of CounterNet in optimally balancing the cost-invalidity trade-off.

Adversarial Robustness. Next, we illustrate that CounterNet does not suffer from decreased robustness of the predictor network resulting from optimizing for the validity loss $\mathcal{L}_2$ (as shown in Lemma 3.2). We compare the robustness of CounterNet's predictor network $f(\cdot)$ against two baselines: (i) the base predictive model described in Table 2; and (ii) CounterNet without freezing the predictor at the second stage of the optimization (*CounterNet-NoFreeze*).

Figure 4 illustrates the perturbation stability [54] of all three CounterNet variants against adversarial examples (generated via projected gradient descent [25]). CounterNet achieves comparable perturbation stability as the base model, which indicates that CounterNet reaches its robustness upper bound (i.e., the robustness of the base model). Moreover, the empirical results in Figure 4 confirm Lemma 3.2 as *CounterNet-NoFreeze* achieves significantly poorer stability. We observe similar patterns with different attack methods on other datasets (see Appendix C). These results show that by freezing the predictor and encoder networks at the second stage of our coordinate descent procedure, CounterNet suffers less from the vulnerability issue created by the adversarial examples.

Feasibility of CF Explanations. Finally, we show how CounterNet's training procedure can be adapted to ensure that the generated CF examples are feasible. In particular, we attempt to use projected gradient descent during the training of CounterNet and enforce hard constraints during the inference stage in order to ensure that the generated CF examples satisfy immutable feature constraints (e.g., `gender` should remain unchanged). At the training stage, a CF example x' is first generated from $g(\cdot)$, and is projected into its feasible space (i.e., $x'' = \mathbb{P}(x')$). Next, we optimize CounterNet over the prediction $\hat{y}_x$ and its projected CF example x'' (via our block-wise coordinate descent procedure). During inference, we enforce that the set of immutable features remains unchanged.

Table 6 shows that enforcing immutable features (via projected gradient descent) does not negatively impact the validity and proximity of the CF examples. This result shows that CounterNet can produce CF examples that respect feasibility constraints.

Training Time. Figure 5 compares the training time for the base model (described in Table 8) and CounterNet, measured in seconds per epoch. CounterNet requires ~3X longer to train when compared to the base model, which is only optimized for prediction accuracy.

Note that although training time is a useful metric for evaluating the speed of the model, this cost only occurs once, making it a secondary metric in terms of evaluating the overall performance of

Table 5: Ablation analysis of CounterNet. Each ablation leads to degraded performance, which in turn, showcases the significance of various design choices in CounterNet.

Ablation	Adult		Credit		HELOC		OULAD	
	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.
CounterNet-BCE	0.86	.238	0.96	.210	0.86	.238	0.95	.101
CounterNet-SingleBP	0.64	.248	0.92	.251	0.93	.206	0.94	.110
CounterNet-Separate	0.96	.257	0.99	.265	0.91	.161	0.94	.097
CounterNet-NoPass-p_x	0.97	.256	0.99	.339	0.98	.147	0.98	.101
CounterNet-Posthoc	1.00	.276	1.00	.247	1.00	.153	0.99	.099
CounterNet	**1.00**	**.196**	**1.00**	**.132**	**1.00**	**.125**	**1.00**	**.075**

Table 6: Impact of the immutable feature constraints in CounterNet. CounterNet generates feasible CF explanations *without* sacrificing validity and proximity.

Dataset	Val. Diff.	Prox. Diff.
Adult	0.0	.009
Credit	0.0	.005
OULAD	0.0	.004

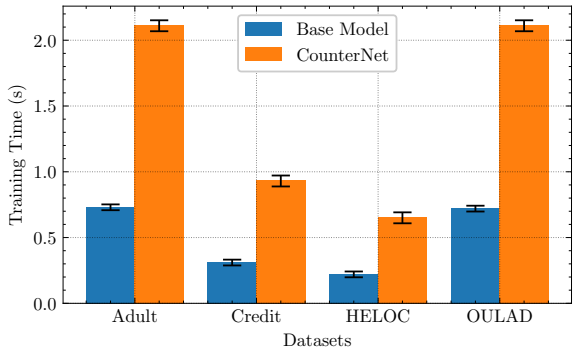


Figure 5: Training time of base MLP model and CounterNet for each epoch (in seconds). The result is obtained on an NVIDIA V100 GPU machine with batch size set to 128. CounterNet takes roughly 3X training time than the base model.

the model. A more critical metric to consider is the inference time for each data point, as a slow inference time directly impacts the usability and deployability of CF explanation techniques (inference time is shown as *runtime* in Table 4).

5 DISCUSSION

Although our experiments exhibit CounterNet's superior performance as compared to post-hoc baselines, these two methods have somewhat different motivations. While post-hoc methods are designed for generating CF explanations for trained black-box ML models (whose training data and model weights might not be available), CounterNet is most suitable for scenarios when the ML model developers aspire to build prediction and explanation modules from scratch, where the training data can be exploited to optimize the generation of CF examples. Due to tighter government regulations (e.g., EU General Data Protection Regulation which enforces the "right to explanation" [53]), it is becoming increasingly important for service providers to provide explanations for any algorithm-mediated decisions. We anticipate CounterNet to be valuable for service providers who wish to comply with GDPR-style regulations without sacrificing their operational effectiveness (e.g., reduced predictive power). Importantly, CounterNet can still be used to interpret proprietary ML models by forcing its predictor network to mimic that proprietary model.

CounterNet has two limitations. (i) First, CounterNet does not consider other desirable aspects in CF explanations, such as diversity [29], recourse cost [48], and causality [18]. Further research is needed to address these issues. (ii) Secondly, although CounterNet is suitable for real-time deployment given its superior performance in its highly aligned CF explanations and speed, one must be aware of the possible negative impacts of its CF explanations to human end-users. It is important to ensure that generated CF examples do not amplify or provide support to the narratives resulting from pre-existing race-based and gender-based societal inequities (among others). One short-term workaround is to have humans in the loop. We can provide CounterNet's explanations as a decision-aid to a well-trained human official, who is in charge of communicating the decisions of ML models to human end-users in a respectful and humane manner. In the long run, further qualitative and quantitative studies are needed to understand the social impacts of CounterNet.

6 CONCLUSION

This paper proposes *CounterNet*, a novel learning framework that integrates predictive model training and CF example generation into a single *end-to-end* pipeline. Unlike prior work, CounterNet ensures that the objectives of predictive model training and CF example generation are closely aligned. We adopt a block-wise coordinate descent procedure to effectively train CounterNet. Experimental results show that CounterNet outperforms state-of-the-art baselines in validity, proximity, and runtime, and is highly competitive in predictive accuracy, sparsity, and closeness to data manifold. In a nutshell, this paper represents a first step towards developing end-to-end counterfactual explanation systems.

REFERENCES

[1] Sushant Agarwal. 2020. Trade-Offs between Fairness and Interpretability in Machine Learning. *Proc. 3rd International Workshop on AI for Social Good* (2020).

[2] Ioannis Arapakis, Souneil Park, and Martin Pielot. 2021. Impact of Response Latency on User Behaviour in Mobile Web Search. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*. 279–283.

[3] Raphael Baena, Lucas Drumetz, and Vincent Gripon. 2022. Preserving Fine-Grain Feature Information in Classification via Entropic Regularization. *arXiv preprint arXiv:2208.03684* (2022).

[4] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2020. Explainable Machine Learning in Deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 648–657. https://doi.org/10.1145/3351095.3375624

[5] Reuben Binns, Max Van Kleek, Michael Veale, Ulrik Lyngs, Jun Zhao, and Nigel Shadbolt. 2018. 'It's Reducing a Human Being to a Percentage' Perceptions of Justice in Algorithmic Decisions. In *Proceedings of the 2018 Chi conference on human factors in computing systems*. 1–14.

[6] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1721–1730.

[7] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: deep learning for interpretable image recognition. In *Advances in neural information processing systems*. 8930–8941.

[8] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations Based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 590–601.

[9] FICO. 2018. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge.

[10] Aritra Ghosh, Himanshu Kumar, and P Shanti Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[11] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[12] Victor Guyomard, Françoise Fessant, Thomas Guyet, Tassadit Bouadi, and Alexandre Termier. 2022. VCNet: A self-explaining model for realistic counterfactual generation. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Matthias Hein and Maksym Andriushchenko. 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems* 30 (2017).

[15] Like Hui and Mikhail Belkin. 2021. Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=hsFN92eQEla

[16] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615* (2019).

[17] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2020. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050* (2020).

[18] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. 2021. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 353–362.

[19] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*. PMLR, 1885–1894.

[20] R Kohavi and B Becker. 1996. UCI Machine Learning Repository: Adult Data Set.

[21] Jakub Kuzilek, Martin Hlosta, and Zdenek Zdrahal. 2017. Open university learning analytics dataset. *Scientific data* 4 (2017), 170171.

[22] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1675–1684.

[23] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. 2013. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 623–631.

[24] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*. 4765–4774.

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJzIBfZAb

[26] Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2019. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277* (2019).

[27] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38.

[28] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. 2020. Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 417–431.

[29] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 607–617.

[30] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. 2019. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592* (2019).

[31] Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. 2022. CounteRGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs. In *Uncertainty in Artificial Intelligence*. PMLR, 1488–1497.

[32] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*. 3126–3132.

[33] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. 2021. *Manipulating and Measuring Model Interpretability*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445315

[34] Kaivalya Rawal, Ece Kamar, and Himabindu Lakkaraju. 2020. Can I Still Trust You?: Understanding the Impact of Distribution Shifts on Algorithmic Recourses. *arXiv preprint arXiv:2012.11788* (2020).

[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[36] Pau Rodríguez, Massimo Caccia, Alexandre Lacoste, Lee Zamparo, Issam Laradji, Laurent Charlin, and David Vazquez. 2021. Beyond trivial counterfactual explanations with diverse valuable explanations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1056–1065.

[37] Alexis Ross, Himabindu Lakkaraju, and Osbert Bastani. 2021. Learning Models for Actionable Recourse. *Advances in Neural Information Processing Systems* 34 (2021).

[38] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[39] Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1756–1764.

[40] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. 2020. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 19655–19666.

[41] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[42] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. 2020. Explanation by Progressive Exaggeration. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1xFWgrFPS

[43] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[45] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. 2021. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access* 9 (2021), 11974–12001.

[46] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*. PMLR, 3319–3328.

[47] Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. 2021. Towards Robust and Reliable Algorithmic Recourse. *Advances in Neural Information*

Processing Systems 34 (2021).

[48] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 10–19.

[49] Arnaud Van Looveren and Janis Klaise. 2019. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584* (2019).

[50] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. *arXiv preprint arXiv:2010.10596* (2020).

[51] Sahil Verma, Keegan Hines, and John P Dickerson. 2022. Amortized generation of sequential algorithmic recourses for black-box models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8512–8519.

[52] Julius Von Kügelgen, Amir-Hossein Karimi, Umang Bhatt, Isabel Valera, Adrian Weller, and Bernhard Schölkopf. 2022. On the fairness of causal algorithmic recourse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9584–9594.

[53] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[54] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. 2021. Do Wider Neural Networks Really Help Adversarial Robustness? *Advances in Neural Information Processing Systems* 34 (2021).

[55] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[56] Fan Yang, Sahan Suresh Alva, Jiahao Chen, and Xia Hu. 2021. Model-Based Counterfactual Synthesizer for Interpretation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1964–1974. https://doi.org/10.1145/3447548.3467333

[57] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2 (2009), 2473–2480.

[58] Yixue Zhao, Marcelo Schmitt Laser, Yingjun Lyu, and Nenad Medvidovic. 2018. Leveraging program analysis to reduce user-perceived latency in mobile applications. In *Proceedings of the 40th International Conference on Software Engineering*. 176–186.

A SUPPLEMENTAL PROOF

A.1 Proof of Lemma 3.1

Proof. $\nabla_\theta \mathcal{L}_1 = \nabla_\theta \|y - \hat{y}_x\|_2$, and $\nabla_\theta \mathcal{L}_2 = \nabla_\theta \|(1 - \hat{y}_x) - \hat{y}_{x'}\|_2$. Then, we have

$$\begin{aligned}\nabla_\theta \mathcal{L}_1 &= \nabla_\theta y^2 - 2y \cdot \nabla_\theta \cdot \hat{y}_x + \nabla_\theta \hat{y}_x^2 \\ &= -2y \cdot \nabla_\theta \hat{y}_x + \nabla_\theta \hat{y}_x^2 \\ &= -2y \cdot \nabla_\theta \hat{y}_x + 2\hat{y}_x \nabla_\theta \hat{y}_x \\ &= 2(\hat{y}_x - y) \cdot \nabla_\theta \hat{y}_x\end{aligned}$$

Since $x' \to x$, as we expect CF example x' is closed to the original instance x, we can replace x' to x in $\mathcal{L}_2$. Then, we have

$$\begin{aligned}\nabla_\theta \mathcal{L}_2 &= \nabla_\theta (1 - 2\hat{y}_x)^2 \\ &= -4 \cdot \nabla_\theta \hat{y}_x + 4 \cdot \nabla_\theta \hat{y}_x^2 \\ &= -4 \cdot \nabla_\theta \hat{y}_x + 4 \cdot 2 \cdot \hat{y}_x \cdot \nabla_\theta \hat{y}_x \\ &= 4 \cdot (2\hat{y}_x - 1) \nabla_\theta \hat{y}_x\end{aligned}$$

Hence,

$$\begin{aligned}\nabla_\theta \mathcal{L}_1 \cdot \nabla_\theta \mathcal{L}_2 &= 2 \cdot (\hat{y}_x - y) \cdot \nabla_\theta \hat{y}_x \cdot 4 \cdot (2\hat{y}_x - 1) \nabla_\theta \hat{y}_x \\ &= 8 \cdot (\hat{y}_x - y) \cdot (2\hat{y}_x - 1) (\nabla_\theta \hat{y}_x)^2\end{aligned}$$

Since $(\nabla_\theta \hat{y}_x)^2 > 0$, we only need to prove whether $(\hat{y}_x - y) \cdot (2\hat{y}_x - 1)$ is positive or negative.

Given that $|\hat{y}_x - y| < 0.5$,

- if $y = 1$, we have $0.5 < \hat{y}_x < 1$. Then, $(\hat{y}_x - y) < 0$, $(2\hat{y}_x - 1) > 0$.
- if $y = 0$, we have $0 < \hat{y}_x < 0.5$. Then, $(\hat{y}_x - y) > 0$, $(2\hat{y}_x - 1) < 0$.

Therefore, $(\hat{y}_x - y) \cdot (2\hat{y}_x - 1) < 0$. Hence, $\nabla_\theta \mathcal{L}_1 \cdot \nabla_\theta \mathcal{L}_2 < 0$.

□

A.2 Proof of Lemma 3.2

Proof. Assuming $f_\theta(x) \to y$ as we expect the predictor network produces accurate predictions, and $y = \{0, 1\}$, we can replace $f_\theta(x)$ to y. Then, minimizing $\mathcal{L}_2$ (in Lemma 3.2) indicates minimizing $\|y - (1 - f_\theta(x'))\|_2$. Since $0 < f_\theta(\cdot) < 1$, we have

$$\min \left\|y - (1 - f_\theta(x'))\right\|_2 = \max \left\|y - f_\theta(x')\right\|_2$$

By replacing y to $f_\theta(x)$, then minimizing $\mathcal{L}_2$ indicates maximizing $\|f_\theta(x) - f_\theta(x')\|_2$. By definition, the lipschitz constant K is

$$K = \sup_{x' \in \mathbb{B}(x,\epsilon)} \{\left\|\nabla f_\theta(x')\right\|_2 = \sup_{x' \in \mathbb{B}(x,\epsilon)} \left\{ \frac{\|f_\theta(x) - f_\theta(x')\|}{\|x - x'\|} \right\}$$

where minimizing $\mathcal{L}_2$ increases $\|f_\theta(x) - f_\theta(x')\|_2$. Therefore, the lipschitz constant K increases.

□

B IMPLEMENTATION DETAILS

Here we provide implementation details of CounterNet and five baselines on four datasets listed in Section 4. The code can be found in the supplemental material.

B.1 Software and Hardware Specification

We use Python (v3.7) with Pytorch (v1.82), Pytorch Lightning (v1.10), numpy (v1.19.3), pandas (1.1.1) and scikit-learn (0.23.2) for the implementations. Our experiments were run on a Debian-10 Deep Learning Image with CUDA 11.0 on the Google Cloud Platform.

The CounterNet network is trained on NVIDIA Tesla V100 with an 8-core Intel machine. CF generation of four baselines is run on a 16-core Intel machine with 64 GB of RAM. The evaluation is generated from the same 16-core machine.

B.2 CounterNet Implementation Details

Across all six datasets, we apply the following same settings in training CounterNet: We initialize the weights as in He et al. [13]. We adopt the Adam with mini-batch size of 128. For each datasets, we trained the models for up to 1×10^3 iterations. To avoid gradient explosion, we apply gradient clipping by setting the threshold to 0.5 to clip gradients with norm above 0.5. We set dropout rate to 0.3 to prevent overfitting. For all six datasets, we set $\lambda_1 = 1.0$, $\lambda_2 = 0.2$, $\lambda_3 = 0.1$ in Equation 2.

The learning rate is the only hyper-parameter that varies across six datasets. From our empirical study, we find the training to CounterNet is sensitive to the learning rate, although a good choice of loss function (e.g. choosing MSE over cross-entropy) can widen the range of an "optimal" learning rate. We apply grid search to tune the learning rate, and our choice is specified in Table 7.

Additionally, we specify the architecture's details (e.g. dimensions of each layer in encoder, predictor and CF generator) in Table 7. The numbers in each bracket represent the dimension of the transformed matrix. For example, the encoder dimensions for adult dataset is [29, 50, 10], which means that the dimension of input $x \in \mathbb{R}^d$ is 29 (e.g. $d = 29$); the encoder first transforms the input into a 50 dimension matrix, and then downsamples it to generate the latent representation $z \in \mathbb{R}^k$ where $k = 10$.

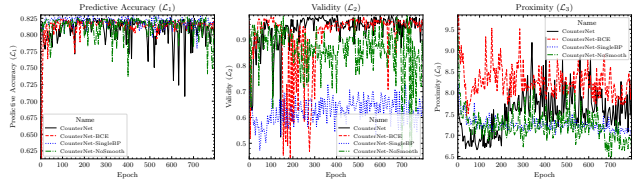
Table 7: Hyperparameters and architectures for each dataset.

Dataset	Learning Rate	Enc. Dims	Pred. Dims	CF Generator Dims
Adult	0.003	[29, 50, 10]	[10, 10, 2]	[20, 50, 29]
Credit	0.003	[33, 50, 10]	[10, 10, 2]	[20, 50, 33]
HELOC	0.005	[35, 100, 10]	[10, 10, 2]	[20, 100, 35]
OULAD	0.001	[127, 200, 10]	[10, 10, 2]	[20, 200, 127]

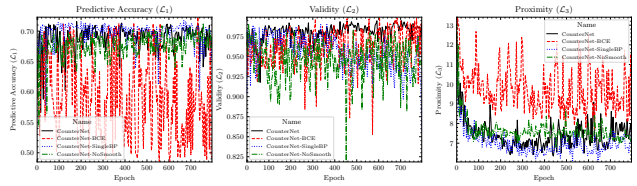
B.3 Hyper-parameters for Baselines

Next, we describe the implementation of baseline methods. For VanillaCF and ProtoCF, we follow author's instruction as much as we can, and implement them in Pytorch. For VanillaCF, DiverseCF and ProtoCF, we run maximum 1×10^3 steps. After CF generation, we convert the results to one-hot-encoding format for each categorical feature. For training the VAE-CF, we follow Mahajan et al. [26]'s settings on running maximum 50 epoches and setting the batch size to 1024. We use the same learning rate as in Table 7 for VAE training.

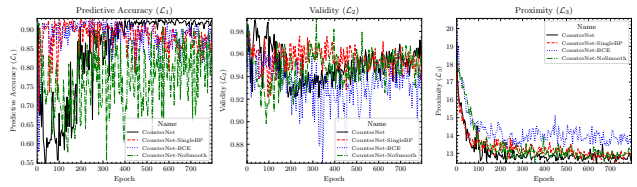
For training predictive models for baseline algorithms, we apply grid search for tuning the learning rate, which is specified in Table 8.



(a) Learning curves of model ablations on the Adult dataset.



(b) Learning curves of model ablations on the HELOC dataset.



(c) Learning curves of model ablations on the OULAD dataset.

Figure 7: Learning curves of $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ of model ablations on the Adult (7a), HELOC (7b), and OULAD (7c) dataset.

Table 8: Learning rate of the predictive models on each dataset.

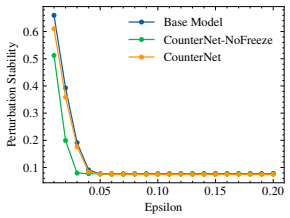
Dataset	Learning Rate
Adult	0.01
Credit	0.003
HELOC	0.005
OULAD	0.001

Similar to training the CounterNet, we adopt the Adam with mini-batch size of 128, and set the dropout rate to 0.3. We train the model for up to 100 iterations with early stopping to avoid overfittings.

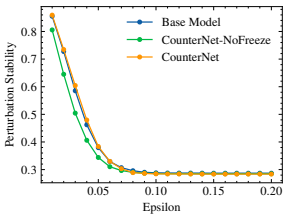
Here, we provide additional results of experiments in Section 4. These results further demonstrate the effectiveness of CounteNet.

C ADDITIONAL ROBUSTNESS RESULTS

We provide supplementary results on evaluating the robustness of the predictor network on three large datasets (i.e., Adult, HELOC and OULAD). In particular, we implement the PGD [25] attack for testing the robustness of the predictive models. Figure 6 illustrates that CounterNet achieves comparable perturbation stability (i.e., the robustness of the predictive model) as the base model. In addition, Figure 6 supports the findings in Lemma 3.2 since *CounterNet-NoFreeze* consistently achieves lower stability than base models and CounterNet.



(a) Robustness of $f(\cdot)$ under PGD attack on OULAD dataset.



(b) Robustness of $f(\cdot)$ under PGD attack on HELOC dataset.

Figure 6: Robustness of $f(\cdot)$ under the PGD [25] attack.

D ABLATIONS ON COUNTERNET'S TRAINING

In addition, we provide supplementary results on ablation analysis of three large datasets (Adult, HELOC, and OULAD) to understand the design choices of the CounterNet training (shown in Figure 7). This figure shows that compared to CounterNet's learning curve for $\mathcal{L}_2$, *CounterNet-BCE* and *CounterNet-NoSmooth*'s learning curves show significantly higher instability, illustrating the importance of MSE-based loss functions and label smoothing techniques. Moreover, *CounterNet-SingleBP*'s learning curve for $\mathcal{L}_2$ performs poorly in comparison, which illustrates the difficulty of optimizing three divergent objectives using a single BP procedure. In turn, this also illustrates the effectiveness of our block-wise coordinate descent optimization procedure in CounterNet's training. These results show that all design choices made in Section 3 contribute to training the model effectively.

E IMPACT OF NEURAL NETWORK STRUCTURES

We further study the impact of the different neural network blocks. In our experiment, we primarily use multi-layer perception as it is a suitable baseline model for the tabular data. For comparison, We also implemented the CounterNet with Convolutional building blocks (i.e. replace the feed forward neural network with convolution layer). We implemented the convolutional CounterNet on the Adult dataset. To train the feed forward neural network with convolution layers, we set the learning rate as 0.03 and $\lambda_1 = 1.0$, $\lambda_2 = 0.4$, $\lambda_3 = 0.01$. The rest of the configuration is exactly the same as training CounterNet with MLP.

Table **??** shows comparison between CounterNet with convolutional building blocks (*CounterNet-Conv*) and multi-layer perceptions (*CounterNet-MLP*). The results indicate that CounterNet-Conv matches the performances of CounterNet-MLP. In fact, CounterNet-Conv performs slightly worse than CounterNet-MLP because the convolutional block is not well-suitable for tabular datasets. Yet, CounterNet-Conv outperforms the rest of our post-hoc baselines in validity (with reasonably good proximity score). This illustrates CounterNet's potential real-world usage in various settings as it is agnostic to the network structures.