CounterNet: End-to-End Training of Counterfactual Aware Predictions

Hangzhi Guo [1] **Thanh Hong Nguyen** [2] **Amulya Yadav** [1]

Abstract

This work presents *CounterNet*, a novel *end-to-end* learning framework which integrates the predictive model training and the counterfactual (CF) explanation into a single end-to-end pipeline. Prior CF explanation techniques rely on solving separate time-intensive optimization problems to find CF examples for every single input instance, and also suffer from the misalignment of objectives between model predictions and explanations, which leads to significant shortcomings in the quality of CF explanations. CounterNet, on the other hand, integrates both prediction and explanation in the same framework, which enable the optimization of the CF example generation only *once* together with the predictive model. We propose a novel variant of back-propagation which can help in effectively training CounterNet's network. Finally, we conduct extensive experiments on multiple real-world datasets. Our results show that CounterNet generates high-quality predictions, and corresponding CF examples (with high validity) for any new input instance significantly faster than existing state-of-the-art baselines.

1. Introduction

From an end-user perspective, counterfactual (CF) explanation [1] techniques (Wachter et al., 2017) are more preferable to other widely known approaches, such as feature-based explanation (e.g., LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017)). 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 x', which is closest possible to x but gets a different (or opposite) prediction from the ML model.

Unfortunately, existing CF explanation techniques are ill-suited for real-world deployment due to three major limitations. First, existing CF explanation techniques search for CF examples by solving an optimization problem for each input instance separately (Wachter et al., 2017; Mothilal et al., 2020; Ustun et al., 2019). This optimization problem is computationally intensive, which is not viable in low-resource and time-constrained environments. Second, existing techniques are post-hoc in nature, i.e., they assume a fixed black-box ML model. As a result, the training procedure of this black-box ML model is completely uninformed by the optimization procedure that finds CF examples in existing techniques. Unfortunately, this causes the generated counterfactuals to align poorly with the original black-box ML model, leading to shortcomings in the validity of the generated CF examples (we illustrate this in Section A). Finally, it is important to ensure that the generated CF examples differ from the original input instance in as few features as possible, so that the end-user can be provided an explanation in fewer features. Unfortunately, prior work mainly focuses on minimizing the distance between the input instance and CF example (by applying L_1-norm "sparsity" terms to their loss function), as opposed to minimizing the number of features with differences.

In this paper, we propose *CounterNet*, a novel learning framework that combines the training of the ML predictive model and the generation of corresponding CF examples into a single end-to-end (i.e., from input to prediction to explanation) pipeline. CounterNet addresses the limitations of existing CF techniques via four key contributions. First, unlike conventional post-hoc approaches, CounterNet uses a (neural network) model-based CF generation method, enabling the joint training of its CF generation network and the predictor network. This joint training is key to better alignment between CF explanation and prediction, leading to superior performance (as we show in Section A). Second, to overcome challenges in effectively training CounterNet, we propose a new loss function formulation, a new variant of back-propagation procedure, and use label smoothing techniques to stabilize CounterNet's network training process. By adopting this end-to-end learning process, CounterNet

[1]The Pennsylvania State University, State Collage, PA, USA [2]University of Oregon, Eugene, OR, USA. Correspondence to: Hangzhi Guo <hangz@psu.edu>, Amulya Yadav <amulya@psu.edu>.

ICML (International Conference on Machine Learning) Workshop on Algorithmic Recourse.

[1]Counterfactual explanation is also closely related to recourse (Ustun et al., 2019) and contrastive explanation (Dhurandhar et al., 2018). Although these terms are proposed under different contexts, their differences to CF explanation has been blurred (Verma et al., 2020), i.e. these terms are used interchangeably.

can generate CF examples with high validity while ensuring that its predictor network also produces highly accurate predictions. Third, we introduce the *robustness* as an important metric to evaluate the real-world usability of CF explanation techniques, which stems from a desire to present explanations to end-users in terms of fewer feature differences (between the input instance and the CF example).

Finally, we conduct a rigorous experimental evaluation of CounterNet - our analysis shows that CounterNet significantly outperforms state-of-the-art CF explanation techniques. In particular, CounterNet is the only method that consistently achieves more than 97% validity and 98% robustness among its generated CF examples. Further, CounterNet runs orders of magnitude faster than state-of-the-art baselines on a wide variety of real-world datasets.

2. Related Work

Our work is closely related to prior literature on counterfactual explanation techniques, which finds new instances that lead to different predicted outcomes. Wachter et al. (2017) propose VanillaCF which generates CF examples 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 Wachter et al. (2017)'s method, optimize other aspects, such as recourse cost (Ustun et al., 2019), fairness (Sharma et al., 2020), diversity (Mothilal et al., 2020), closeness to the data manifold (Van Looveren & Klaise, 2019), and causal constraints (Mahajan et al., 2019). Previous studies (Binns et al., 2018; Bhatt et al., 2020) show that counterfactual explanations can benefit end-users to make decisions. Unfortunately, most prior work follows a post-hoc approach (explaining after a trained ML model is provided) , which is ill-suited for real-world deployment. This post-hoc approach is not only time-consuming, but is also exposed to the issue of misalignment between the predictive model training and the counterfactual example generation (as we show in Section A). In our work, we propose CounterNet, an end-to-end framework which overcomes these limitations to efficiently generate accurate predictions, and corresponding highly-aligned CF explanations.

3. The Proposed Framework: CounterNet

Unlike prior work, our proposed framework CounterNet relies on a novel integrated architecture which combines the predictive model training and the counterfactual example generation into a single optimization framework. This integration allows us to optimize the CF example generation component only once together with the predictive model training component (as part of a single neural network architecture). Through this integration, we can simultaneously optimize the accuracy of the trained predictive model and the quality of the generated counterfactual examples. Moreover, the model prediction and the CF example generation both derive from a shared component. This design leads to closer alignment between prediction and explanation.

Formally, given an input instance $x \in \mathbb{R}^d$, CounterNet aims to generate two outputs: (i) the ML prediction component generates a predicted value $\hat{y}_x$ for input instance x; and (ii) the CF generation component produces a CF example $x' \in \mathbb{R}^d$ as an explanation for input instance x. Ideally, the counterfactual example x' should get a different prediction $\hat{y}_{x'}$ from CounterNet's ML prediction component, as compared to the prediction $\hat{y}_x$ on the original input x (i.e., $\hat{y}_{x'} \neq \hat{y}_x$). Next, we illustrate the framework design.

3.1. Network Architecture

Figure 1 illustrates the architecture of CounterNet. At a high level, CounterNet 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 vector representation is passed through both the predictor network and the CF generator network. The predictor network outputs a softmax representation of the prediction $\hat{y}_x = f(z_x)$, given x. Simultaneously, the CF generator network produces a CF example $x' = g(z_x)$ for input instance x.

Furthermore, to ensure that the CF generator network outputs valid counterfactual examples (i.e., $\hat{y}_x \neq \hat{y}_{x'}$), the output of the CF generator network $x' = g(h(x))$ is also passed back as an input through the encoder and predictor networks when CounterNet is trained. Importantly, the addition of this feedback loop (from the output of CF generator network back into the encoder and predictor networks) enables us to explicitly optimize the *validity* of CF examples output by the CF generator network, i.e., it enables us to train the encoder, predictor and CF generator networks together in a way such that the predictor network outputs opposite predictions $\hat{y}_x$ and $\hat{y}_{x'}$ for the original input instance x and the CF example output by the CF generator network x', respectively. Note that this '*feedback loop*' neuronal connection is only needed at training time, and is removed from the network at test time. Figure 1 shows neuronal connections required during training and test times using solid and dashed arrows, respectively.

Design of Encoder, Predictor & CF Generator. All three components in CounterNet's architecture are composed of multiple fully connected feed-forward neural network layers. While we fix the number of layers inside each component, the number of neurons in each of those layers are hyperparameters, and are found separately for each dataset via grid

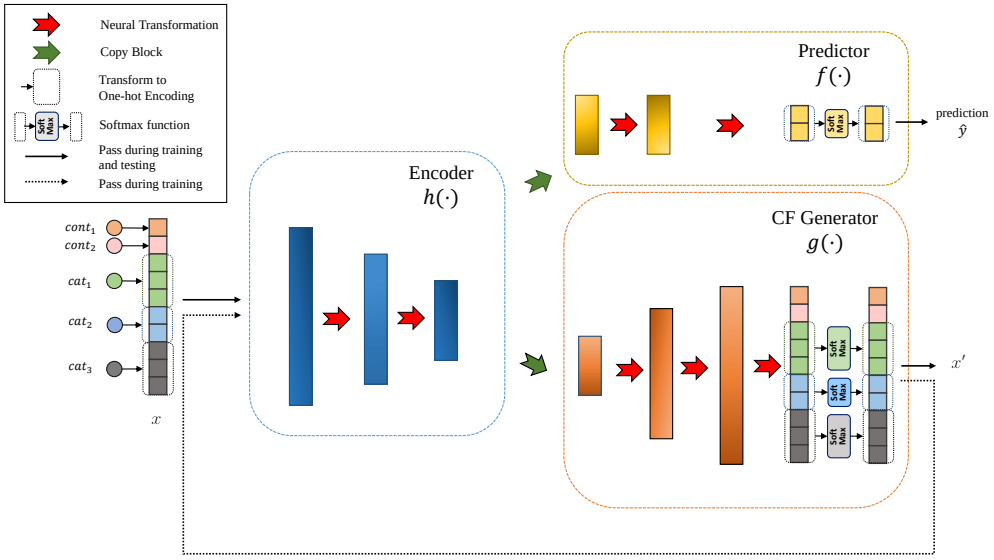

Figure 1. Architecture of CounterNet contains three components: an encoder (blue) that transforms the input into a dense latent vector; a predictor network (yellow) that outputs the prediction; and a CF generator (orange) to produce explanations.

search. The encoder network in CounterNet is composed 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 takes this latent vector representation z as input, and passes it through an additional feed-forward neural network layer. Finally, the predictor network outputs the soft-max probability distribution over predictions with a fully-connected layer. On the other hand, the CF generator network has a flipped structure as compared to the encoder network, where it takes the latent vector representation $z \in \mathbb{R}^k$ as input, and up-samples to generate CF examples $x' \in \mathbb{R}^d$.

Each feed-forward neural network layer inside CounterNet uses LeakyRelu activation functions (Xu et al., 2015). We also apply dropout (Srivastava et al., 2014) after every layer to avoid overfitting. More details are in the Appendix B.

Customizing for 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, and consider them as continuous variables between [0, 1]. In addition, we add a softmax layer after the final output layer in the CF generator network for each categorical feature (Figure 1). This slight adjustment ensures that CounterNet generates CF examples which respect the one-hot encoding format (as the output of the softmax layer will sum up to 1).

3.2. CounterNet Loss Function

Each part of our three-part loss function corresponds to a desirable characteristic that we expect in CounterNet's output: (i) *predictive accuracy* - we expect the predictor network of Figure 1 to output highly accurate predictions; (ii) *counterfactual validity* - we expect that CF examples x' output by the CF generator network are valid, i.e., they get opposite (or unequal) predictions from the predictor network (e.g. $\hat{y}_x \oplus \hat{y}_{x'} = 1$); and (iii) *proximity* - we expect that input instance x can be transformed into CF example x' with minimum modifications to feature values.

Based on these three characteristics, we adopt the three parts of CounterNet's loss function as follows:

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

where N denotes the number of instances in our dataset, $\mathcal{L}_1$ 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, $\mathcal{L}_2$ denotes the MSE between the original predicted label 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 (for each input instance x_i) by ensuring that the predictions on x'_i and x_i are as different as possible. Finally, $\mathcal{L}_3$ represents the average distance between input instances x_i and the counterfactual examples x'_i (averaged across $i = 1 \ldots N$ datapoints), which aims to maximize proximity. This choice of loss functions is key

to CounterNet's superior performance, as replacing $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ with alternate functional forms leads to degraded performance (as we show in Appendix A).

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:

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

where $(\lambda_1, \lambda_2, \lambda_3)$ are hyper-parameters to balance the trade-off between the loss components. Equation 2 is difficult to optimize because of three divergent objectives, which leads to poor convergence of gradient descent. Next, we propose a new variant of backpropagation to remedy this issue.

3.3. Practical Choices for Training

The conventional way of solving the optimization problem in Equation 2 is to use gradient descent with back propagation (BP). However, as we show in Section A, directly optimizing the entire objective in Equation 2 using conventional BP leads to poor quality of prediction and CF example outcomes. This occurs because the optimization problem in Equation 2 consists of three divergent objectives, i.e., $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, each of which individually tries to move the gradient descent based backpropagation search in different directions. As a result, the overall gradient direction (i.e., combined gradient across all three loss components) fluctuates significantly with increasing number of training epochs, which leads to poor convergence of the training procedure.

To remedy this problem, we propose a novel variant of BP. We divide the problem of optimizing Equation 2 into two separate parts: (i) optimizing predictive accuracy; and then (ii) optimizing the validity of CF generation. Note that in the CounterNet architecture, the predictive accuracy of the predictor network is primarily influenced by $\mathcal{L}_1$ and $\mathcal{L}_2$, whereas the CF generation is dictated primarily by $\mathcal{L}_2$ and $\mathcal{L}_3$. We thus propose a double-back BP procedure to iteratively optimize these two parts. Specifically, for each mini-batch of m data points $\{x^{(i)}, y^{(i)}\}^m$, the network updates its weights θ twice through back propagation. For the first update, it computes $\theta' = \theta - \nabla_\theta(\lambda_1 \cdot \mathcal{L}_1)$, and for the second update, it computes $\theta'' = \theta' - \nabla_{\theta'}(\lambda_2 \cdot L_2 + \lambda_3 \cdot \mathcal{L}_3)$. Thus, we distribute the objective function (Equation 2) into two stages, which lessens the difficulty in training. As we show in Appendix A, this novel variant of double-back BP enables significantly better convergence in training as compared to the conventional BP algorithm.

Finally, inspired from Salimans et al. (2016), we also implement the label smoothing trick to further stabilize our training. Intuitively, label smoothing prevents neural networks from becoming overconfident in their predictions (Müller et al., 2019). Specifically, for each mini-batch $\{x^{(i)}, y^{(i)}\}^m$, we set each $y \sim \mathcal{U}(0.8, 0.95)$ for positive labels, and $y \sim \mathcal{U}(0.05, 0.2)$ for negative labels.

4. Experimental Evaluation

We conduct a comprehensive evaluation of CounterNet's performance against several state-of-the-art baseline CF explanation methods on a wide variety of real-world datasets. The simulated results show that CounterNet achieves highly superior performance over baseline CF explanation techniques. First, CounterNet can consistently generate CF examples with more than 97% validity, whereas baseline methods perform poorly on specific datasets. Second, CounterNet's high CF validity comes with minimal cost in terms of lowered predictive accuracy and proximity. Finally, CounterNet runs significantly faster than post-hoc baseline methods as it runs orders of magnitude faster than the baselines methods. We elaborate the experimental evaluation in Appendix A.

5. Limitations & Conclusion

CounterNet has three limitations. First, as an end-to-end pipeline, CounterNet is less flexible than existing post-hoc CF explanation methods; e.g., CounterNet is not designed to interpret a trained black-box model (as is the case with many existing post-hoc methods). Furthermore, CounterNet does not consider other desirable aspects in CF explanations, such as diversity (Mothilal et al., 2020), recourse cost (Ustun et al., 2019), and fairness (Sharma et al., 2020). Further research is needed to address these issues. Finally, although CounterNet is suitable for real-time deployment given its superior performance in validity and speed, one must be aware of the possible negative impacts to the 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 human 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.

This paper proposes *CounterNet*, a novel learning framework which 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, which leads to superior performance. We also propose a novel BP variant to stabilize CounterNet's training process. Experimental analysis on several real-world datasets shows that CounterNet significantly outperforms state-of-the-art baselines in validity, robustness, and runtime, and is highly competitive in predictive accuracy.

References

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F., and Eckersley, P. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pp. 648–657, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3375624. URL https://doi.org/10.1145/3351095.3375624.

Binns, R., Van Kleek, M., Veale, M., Lyngs, U., Zhao, J., and Shadbolt, N. '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*, pp. 1–14, 2018.

Blake, C. Uci repository of machine learning databases. *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1998.

Cortez, P. and Silva, A. M. G. Using data mining to predict secondary school student performance. 2008.

Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., and Das, P. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 590–601, Red Hook, NY, USA, 2018. Curran Associates Inc.

FICO. Explainable machine learning challenge. https://community.fico.com/s/explainable-machine-learning-challenge, September 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaggle. Titanic - machine learning from disaster. https://www.kaggle.com/c/titanic/overview, September 2018.

Kuzilek, J., Hlosta, M., and Zdrahal, Z. Open university learning analytics dataset. *Scientific data*, 4:170171, 2017.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pp. 4765–4774, 2017.

Mahajan, D., Tan, C., and Sharma, A. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

Mothilal, R. K., Sharma, A., and Tan, C. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617, 2020.

Müller, R., Kornblith, S., and Hinton, G. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Ribeiro, M. T., Singh, S., and Guestrin, C. " 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*, pp. 1135–1144, 2016.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Sharma, S., Henderson, J., and Ghosh, J. Certifai: A common framework to provide explanations and analyse the fairness and robustness of black-box models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pp. 166–172, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371100. doi: 10.1145/3375627.3375812. URL https://doi.org/10.1145/3375627.3375812.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Ustun, B., Spangher, A., and Liu, Y. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 10–19, 2019.

Van Looveren, A. and Klaise, J. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

Verma, S., Dickerson, J., and Hines, K. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Dataset	Size	#Continuous	#Categorical
Adult	32,561	2	6
Student	649	2	14
Titanic	891	2	24
HELOC	10,459	21	2
OULAD	32,593	23	8
Breast Cancer	569	30	0

Table 1. Summary of Datasets used for Evaluation

A. Experimental Evaluation

A.1. Experiment Setup

Baselines. We compare CounterNet against four state-of-the-art post-hoc CF explanation techniques: (i) *VanillaCF* (Wachter et al., 2017) — this post-hoc CF generation method generates CF examples by optimizing counterfactual validity and proximity; (ii) *DiverseCF* and (iii) *ProtoCF* — these two baselines generate CF examples by optimizing for diversity and consistency with prototypes, respectively; and (iv) *VAE-CF* (Mahajan et al., 2019) — this is the state-of-the-art model-based post-hoc approach to CF example generation. VAE-CF is the only baseline which relies on a neural network model to generate CF examples, while the other baselines (VanillaCF, DiverseCF and ProtoCF) generate CF examples by solving explicit optimization problems.

All four baseline techniques require a trained predictive model as input (unlike CounterNet). Thus, for each dataset, we train a neural network model and use it as the base predictive model for all baselines. For fair comparison, our base predictive model is trained in the following manner: (i) we discard the CF generator network inside CounterNet's architecture (Figure 1), and only keep the encoder and predictor networks (since only these two networks are needed by CounterNet to make predictions); (ii) this combination of encoder and predictor networks is solely optimized for predictive accuracy (i.e., $\mathcal{L}_1$), and this trained model is then used as our base predictive model with all baselines. For each dataset, hyperparameter tuning was conducted separately using grid search. The best hyperparameter values for the trained network are included in the appendix.

Datasets. We evaluate CounterNet on six real-world binary classification datasets from diverse domains (Table 1) . Our primary evaluation is done on three large-sized datasets: (i) *Adult* (Blake, 1998) which aims to predict whether an individual's income reaches \$50K, based on their demographic information; (ii) *HELOC* (FICO, 2018) which uses financial information in credit reports to predict whether a homeowner qualifies for a line of credit; and (iii) *OULAD* (Kuzilek et al., 2017) which aims to predict whether MOOC students drop out, based on their online learning logs.

In addition, we experiment with three small-sized datasets: (i) *Breast Cancer Wisconsin (Diagnostic)* (Blake, 1998) which identifies patients with malignant/benign tumors; (ii) *Student Performance* (Cortez & Silva, 2008) which predicts the student grade based on answers to survey questionnaires; and (iii) *Titanic* (Kaggle, 2018) which aims to predict if passengers survived the Titanic shipwreck.

Evaluation Metrics. For each input data point x, CF explanation techniques (including CounterNet) generate two outputs: (i) a prediction $\hat{y}_x$; and (ii) a CF example x', which receives an opposite prediction $\hat{y}_{x'}$ (for binary classification problems, $\hat{y}_x \oplus \hat{y}_{x'} = 1$). We evaluate the quality of both these outputs using separate metrics. For evaluating predictions, we use *predictive accuracy* (since all six datasets are fairly class-balanced). For evaluating CF examples, we use three separate metrics: (i) *Validity*, is defined as the fraction of input data points (in the test set) on which CF explanation techniques output valid counterfactual examples, i.e., the fraction of input data points for which $\hat{y}_x \oplus \hat{y}_{x'} = 1$. (ii) *Proximity*, is defined as the L_1 norm distance between the input x and the generated CF x'. (iii) Finally, we propose a novel metric, called *Robustness*, which measures the robustness of the generated CF example against small manipulations. Formally, let $x = \{x_1, x_2, .., x_d\}$ and $x' = \{x'_1, x'_2, .., x'_d\}$ be the features of the input data point and the CF example, respectively. To measure robustness of CF x', we use a threshold of b, and create a new data point $x'' = \{l_i = \mathbb{1}_{|x_i - x'_i| \leq b} x_i + \mathbb{1}_{|x_i - x'_i| > b} x'_i \; \forall i \in 1 \ldots d\}$, i.e., we replace all features $i \in \{1, d\}$ in x' (with corresponding feature values in x) for which $|x_i - x'_i| \leq b$. Intuitively, x'' represents a slightly altered version of our CF example x' in which we neglect feature differences that are less than b. We repeat this procedure for every input data point (in the test set) and its corresponding CF example. Finally, robustness is defined as the fraction of input data points x on which x'' also remains a valid counterfactual example.

Rationale of Metrics. The need for high *predictive accuracy* in any CF explanation technique is obvious - we do not want to provide explanations for incorrect predictions made by an ML model. For the quality of CF examples, (i) high *validity* is desirable, as it implies the technique's greater effectiveness at creating valid counterfactual examples; (ii) high *proximity* is desirable, as it implies fewer modifications that need to be made to the input data point to convert it into a valid CF example; and finally (iii) high *robustness* is desirable for any CF explanation technique. Intuitively, techniques with high robustness enable us to create end-user explanations only in terms of features in x which differ significantly from corresponding feature values in x' while completely neglecting small feature differences (less than b). This refers to the modified counterfactual example x'' (which neglects small feature differences between x and x') remaining as a valid counterfactual. Thus, high robustness aids interpretability from an end-user perspective.

A.2. Evaluating CounterNet Performance

In short, Tables 2 and 3 show that CounterNet achieves highly superior performance over baseline CF explanation techniques. First, CounterNet can consistently generate CF examples with more than 97% validity (whereas baseline methods perform poorly on specific kinds of datasets). In particular, VanillaCF uses a post-hoc CF generation method very similar to CounterNet; yet, it achieves 17% lower validity on datasets with large number of categorical features. This illustrates the misalignment between the objectives of predictive model training and CF example generation inside post-hoc methods like VanillaCF. On the other hand, CounterNet's joint training of its predictor and CF generation networks ensures that the objectives of these two components are closely aligned, which leads to superior validity of CF examples. Second, the high validity of CounterNet's CF examples is achieved at minimal cost (in terms of lowered predictive accuracy), e.g., CounterNet's predictor network is only 0.5% less accurate than the base MLP network (averaged across all six datasets) which is solely optimized for predictive accuracy. Finally, CounterNet runs in a fraction of time taken by its competitors - it is 3X faster than its nearest competitor and is orders of magnitude faster than the rest of the baselines. We next elaborate our results.

Counterfactual Validity. Table 2 compares the validity of CF examples generated by CounterNet and other baselines on all six datasets. This table shows that CounterNet achieves over 97% validity on all six datasets. In particular, CounterNet achieves $\sim$30%, $\sim$21%, $\sim$ 0.5% higher validity than VanillaCF (its closest competitor) on the Adult, Student, and Titanic datasets, respectively. On the other hand, the performance of CounterNet (and other baselines) is fairly saturated on the HELOC, OULAD and Breast Cancer datasets, e.g., CounterNet achieves an average validity (across these three datasets) of $\sim$99%, whereas VanillaCF achieves a validity score of 100% on these three datasets.

Note that the first three datasets (Adult, Student and Titanic) have disproportionately more categorical features as compared to continuous features, whereas the last three datasets (HELOC, OULAD and Breast Cancer) have disproportionately more continuous features as compared to categorical features (see Table 1). Thus, the results in Table 2 illustrate two things: (i) CounterNet consistently achieves more than 97% validity on all six datasets; (ii) while baseline methods achieve high validity on datasets which contain disproportionate number of continuous features (e.g., they outperform CounterNet by $\sim$1% on these datasets), they achieve very poor validity on the first three datasets (e.g., CounterNet outperforms the best performing baseline by 17% on these datasets). These results highlight the difficulty of achieving high validity with categorical features, and shows that baseline methods are ill-suited for these tasks.

Counterfactual Proximity & Robustness. Table 2 compares the proximity/robustness achieved by CounterNet and baselines on all six datasets. We use a fixed threshold $b = 2$ to compute robustness for all datasets. This table shows that on the Breast Cancer dataset, CounterNet achieves 41% higher robustness as compared to VanillaCF (the next best performing baseline). On all other datasets, the robustness achieved by all methods is fairly saturated, e.g., CounterNet achieves 100% robustness on the Adult, HELOC, Student, and Titanic datasets. Note that on the Titanic dataset, VanillaCF and ProtoCF have N.A. values, as they did not generate any CF examples with feature differences less than $b = 2$.

In terms of proximity, CounterNet is highly competitive against VAE-CF, the only other model-based approach to generating CF examples. Across all six datasets, the difference between the proximity achieved by CounterNet and VAE-CF is 1.8%. However, VanillaCF, DiverseCF, and ProtoCF perform better than CounterNet and VAE-CF in terms of proximity. This is expected behavior since, by design, VanillaCF, DiverseCF and ProtoCF optimize the proximity of each data point separately, whereas VAE-CF and CounterNet globally optimize the proximity across all data points.

Predictive Accuracy. Table 3 compares the predictive accuracy achieved by CounterNet against the base prediction model used with our baseline methods. This table shows that CounterNet exhibits highly competitive performance in terms of predictive accuracy on all six datasets. CounterNet achieves marginally better accuracy on the Student, Titanic and HELOC datasets (row 2,3 & 4), and achieves marginally lower accuracy on the remaining datasets. Across all six datasets, the difference between the predictive accuracies of CounterNet and the base model is $\sim$ 0.5%. This table illustrates that potential benefits achieved by CounterNet's joint training of predictor and CF generator networks do not come at a cost (in terms of reduced predictive accuracy), as CounterNet achieves highly competitive performance against an ML model solely optimized for predictive accuracy.

Runtime Comparison. Table 2 shows the average runtime (in milliseconds) taken by CounterNet and other baselines in generating a CF example for a single data point in the test data. This table shows CounterNet generates CF examples $\sim$3X (on average) faster than VAE-CF, which is the fastest baseline method. Moreover, CounterNet runs three orders of magnitude faster than the other baselines, out of which DiverseCF runs slowest, followed by ProtoCF and VanillaCF (with average runtime across six datasets of 4592, 2328, and 1576 milliseconds, respectively). This result illustrates CounterNet's ability to be used in time-constrained environments such as smartphones and edge ML devices.

Ablation Analysis. We analyze the importance of design choices inside CounterNet on the resulting interpretability/predictive accuracy. We analyze three ablations of Coun-

Datasets	Metrics	Methods				
		VanillaCF	DiverseCF	ProtoCF	VAE-CF	CounterNet
Adult	Validity	0.758	0.535	0.589	0.664	**0.986**
	Proximity	**5.804**	8.002	7.169	8.627	7.230
	Robustness	**1.000**	0.997	0.996	0.994	**1.000**
	Running time	1821.737	6307.111	2492.588	3.037	**1.063**
Student	Validity	0.803	0.527	0.325	0.491	**0.975**
	Proximity	**13.213**	18.909	16.305	21.413	21.334
	Robustness	0.990	0.987	0.987	0.993	**1.000**
	Running time	2558.077	9176.334	2609.500	3.650	**1.657**
Titanic	Validity	0.977	0.524	0.762	0.376	**0.982**
	Proximity	**16.259**	18.137	16.458	19.661	17.238
	Robustness	N.A.	**1.000**	N.A.	**1.000**	**1.000**
	Running time	3213.318	1250.652	2478.933	4.188	**2.282**
HELOC	Validity	**1.000**	0.904	0.998	**1.000**	0.995
	Proximity	5.403	**5.230**	5.879	7.727	6.478
	Robustness	**1.000**	0.992	**1.000**	**1.000**	**1.000**
	Running time	1572.875	4836.099	2508.052	2.682	**0.765**
OULAD	Validity	**1.000**	0.687	**1.000**	**1.000**	0.971
	Proximity	12.842	14.821	13.644	14.576	**12.353**
	Robustness	**1.000**	0.923	**1.000**	**1.000**	0.998
	Running time	2138.736	7110.721	2570.173	3.119	**1.215**
Breast Cancer	Validity	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Proximity	1.422	1.007	**0.729**	1.410	2.410
	Robustness	0.699	0.195	0.294	0.216	**0.986**
	Running time	1270.284	4058.734	2415.588	2.444	**0.537**

Table 2. Evaluation of Counterfactual Examples

Dataset	Base Model	CounterNet
Adult	0.831	0.825
Student	0.908	0.920
Titanic	0.816	0.821
HELOC	0.717	0.718
OULAD	0.934	0.912
Breast Cancer	0.972	0.951

Table 3. Evaluation of CounterNet's Predictive Accuracy

terNet, each of which is created by replacing a specific model component in CounterNet with an alternate choice of component. First, we highlight the importance of the MSE loss functions used to optimize the CounterNet network (Equation 1) by replacing the MSE based $\mathcal{L}_1$ and $\mathcal{L}_2$ loss in Eq. 1 with cross entropy loss (*CounterNet-BCE*). Second, we highlight the importance of CounterNet's novel optimization algorithm (i.e., double-back BP procedure) by using conventional one-pass-through backpropagation optimization to train CounterNet instead (*CounterNet-SingleBP*). Finally, we highlight the importance of label smoothing by excluding it from CounterNet (*CounterNet-NoSmooth*).

Figure 2 compares the individual learning curves for the $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ loss functions achieved by CounterNet and our three ablation models. This figure shows that compared to the original CounterNet's learning curve for the $\mathcal{L}_2$ loss function, *CounterNet-BCE* and *CounterNet-NoSmooth*'s learning curves are characterized by significantly higher instability, which illustrates the importance of our MSE based loss function and label smoothing techniques in CounterNet's effective performance. Moreover, *CounterNet-SingleBP*'s learning curve for the $\mathcal{L}_2$ loss function performs very poorly in comparison, which illustrates the difficulty of simultaneously optimizing three divergent loss functions inside CounterNet using a single backpropagation procedure. In turn, this also illustrates the importance of our double-back BP procedure in CounterNet's effective performance. These results demonstrate that all design choices made in Section 3 contribute to effectively training a high-performance model.

Real-World Usage. We now illustrate how CounterNet can generate interpretable explanations for human end-users. Figure 3 shows an actual data point x from the Adult dataset, and the corresponding valid CF example x' generated by CounterNet. This figure shows that x and x' only differ in three features (Age, Hours/week, and Education). Instead of generating an explanation to the end-user in terms of all three features, CounterNet generates x'' by ignoring feature differences that are less than threshold $b = 2$ (in practice, domain experts can help identify realistic values of b). Note that due to CounterNet's high levels of robustness, x'' also

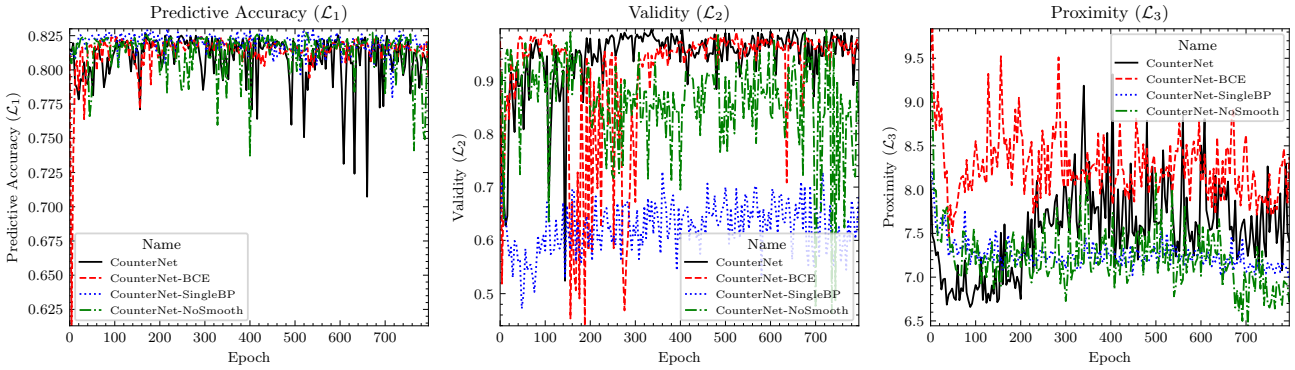

Figure 2. Learning curves of $\mathcal{L}_1$ (left), $\mathcal{L}_2$ (mid), and $\mathcal{L}_3$ (right) of model ablations on the Adult dataset.

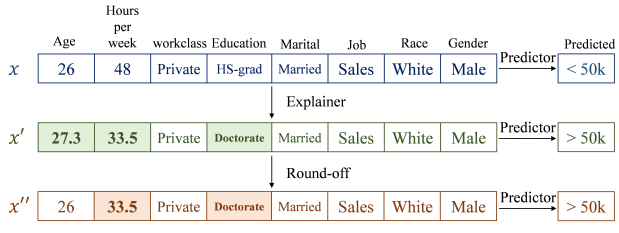

Figure 3. An example CF explanation generated by CounterNet.

remains a valid CF example. After this post-processing step, x and x'' differ in exactly two features (Hours/week and Education), and the end-user is provided with the following natural-language explanation: "*If you want the ML model to predict that you will earn more than US$50K, change your education from* ***HS-Grad*** *to* ***Doctorate****, and reduce the number of hours of work/week from* ***48*** *to* ***33.5****.*"

B. Implementation Details

Here we provide implementation details of CounterNet and four baselines on six datasets listed in Appendix A. The code is available at `https://github.com/BirkhoffG/CounterNet`.

B.1. Software and Hardware Specification

We use Python (v3.7) with Pytorch (v1.60), Pytorch Lightning (v1.10), numpy (v1.19.3), pandas (1.1.1) and scikit-learn (0.23.2) for the implementations. All our experiments were run on a Debian-10 Linux-based 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 are run on a 16-core Intel machine with 64 GB of RAM. The evaluation (results in Table 3 & 2 in the original paper) are generated from the same 16-core machine.

B.2. CounterNet Details

Across all six datasets, we apply the following same settings in training CounterNet: We initialize the weights as in He et al. (2016). 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 $\lambda_1 = 1.0$, $\lambda_2 = 0.01$, $\lambda_3 = 1.0$ 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 5.

Additionally, we specify the architecture's details (e.g. dimensions of each layer in encoder, predictor and CF generator) in Table 5. 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 $k = 10$.

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 $1 \times$

Dataset	Learning Rate
Adult	0.01
Student	0.01
Titanic	0.01
HELOC	0.005
OULAD	0.001
Breast Cancer	0.001

Table 4. Learning rate of the predictive models for each dataset

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. (2019)'s settings on running maximum 50 epoches and setting the batch size to 1024. We use the same learning rate as in Table 5 for VAE training.

For training predictive models for baseline algorithms, we apply grid search for tuning the learning rate, which is specified in Table 4. 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.

Dataset	Learning Rate	Encoder Dims	Predictor Dims	CF Generator Dims
Adult	0.01	[29, 50, 10]	[10, 10, 2]	[10, 50, 29]
Student	0.01	[85, 100, 10]	[10, 10, 2]	[10, 100, 85]
Titanic	0.01	[57, 100, 10]	[10, 10, 2]	[10, 100, 57]
HELOC	0.005	[35, 100, 10]	[10, 10, 2]	[10, 100, 35]
OULAD	0.001	[127, 200, 10]	[10, 10, 2]	[10, 200, 127]
Breast Cancer	0.001	[30, 50, 10]	[10, 10, 2]	[10, 50, 30]

Table 5. Hyperparameters and architectures for each dataset