Countering Overfitting with Counterfactual Examples

Flavio Giorgi
giorgi@di.uniroma1.it
Sapienza University of Rome
Rome, Italy

Fabiano Veglianti
fabiano.veglianti@uniroma1.it
Sapienza University of Rome
Rome, Italy

Fabrizio Silvestri
fsilvestri@diag.uniroma1.it
Sapienza University of Rome, Rome & ISTI-CNR, Pisa
Italy

Gabriele Tolomei
tolomei@di.uniroma1.it
Sapienza University of Rome, Rome & ISTI-CNR, Pisa
Italy

Abstract

Overfitting is a well-known issue in machine learning that occurs when a model struggles to generalize its predictions to new, unseen data beyond the scope of its training set. Traditional techniques to mitigate overfitting include early stopping, data augmentation, and regularization. In this work, we demonstrate that the degree of overfitting of a trained model is correlated with the ability to generate *counterfactual examples*. The higher the overfitting, the easier it will be to find a valid counterfactual example for a randomly chosen input data point. Therefore, we introduce CF-Reg, a novel regularization term in the training loss that controls overfitting by ensuring enough margin between each instance and its corresponding counterfactual. Experiments conducted across multiple datasets and models show that our *counterfactual regularizer* generally outperforms existing regularization techniques.

CCS Concepts

• **Computing methodologies** → **Regularization**; **Machine learning**; **Supervised learning**.

Keywords

Overfitting, generalizability, regularization, counterfactual examples, counterfactual explanations

ACM Reference Format:
Flavio Giorgi, Fabiano Veglianti, Fabrizio Silvestri, and Gabriele Tolomei. 2026. Countering Overfitting with Counterfactual Examples. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770854.3780243

1 Introduction

One of the key challenges in machine learning is developing models that can generalize their predictions to new, unseen data beyond the scope of the training set. When the predictive accuracy of a model on the training set far exceeds that on the test set, this indicates a phenomenon known as *overfitting*. In general, the impact of overfitting is more pronounced for highly complex models like recent deep neural networks with billions of parameters. To compensate for the risk of overfitting, these models require massive amounts of training data, which may not always be feasible.

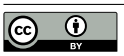
Therefore, several strategies have been proposed in the literature to mitigate the problem of model overfitting. Among these, *early stopping* interrupts the training phase before the model starts learning the noise in the data rather than the actual underlying input/output relationship. Furthermore, *data augmentation* is a technique that artificially increases the training set. For example, in the context of image data, this can include applying translation, flipping, and rotation transformations to input samples. Finally, *regularization* is a collection of training/optimization techniques that try to eliminate irrelevant factors by assessing the importance of features, preventing minor input changes from causing significant output variations.

In this work, we offer an entirely new perspective on model overfitting, establishing a connection with the ability to generate *counterfactual examples* [35]. The notion of counterfactual examples has been successfully used, for instance, to attach post-hoc explanations for predictions of individual instances in the form: "*If A had been different, B would* ***not*** *have occurred*" [30]. Generally, finding the counterfactual example for an instance resorts to searching for the minimal perturbation of the input that crosses the decision boundary induced by a trained model. This task reduces to solving a constrained optimization problem.

In the presence of a strong degree of model overfitting, the decision boundary learned becomes a highly convoluted surface, up to the point where it perfectly separates every training input sample. Therefore, each data point, on average, is "closer" to the decision boundary, making it easier to find the best counterfactual example. The intuition behind this claim is depicted in Figure 1 and grounded in the theoretical foundations of margin theory [38].

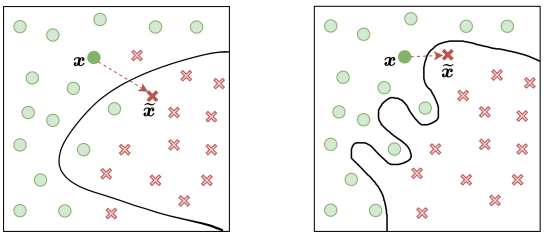
Following this idea, we introduce a novel *counterfactual regularization* term in the training loss that controls overfitting by enforcing a margin between each instance and its hypothetical counterfactual. Below, we summarize the primary novel contributions of our work:

(*i*) We explore the relationship between generalizability and counterfactual explanations, based on margin theory.
(*ii*) We show that counterfactual examples can effectively guide and enhance the model training process.
(*iii*) We propose a counterfactual regularizer (CF-Reg) that outperforms existing regularization techniques.



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2258-5/2026/08
https://doi.org/10.1145/3770854.3780243



(a) No overfitting (b) Overfitting

Figure 1: Distance between an input data point (x) and its counterfactual example ($\tilde{x}$): On average, this may be higher for a well-trained model (a) than an overfitted model (b).

(*iv*) We cast our regularization method in an extensible framework that is compatible with any differentiable counterfactual example generator. The source code for our method is available at the following GitHub repository: https://github.com/hercolelab/CF-Reg.

The remainder of this paper is structured as follows. In Section 2, we summarize related work. Section 3 reviews background concepts, useful for our problem formulation in Section 4. In Section 5, we describe our method that is validated through extensive experiments in Section 6. We discuss the applicability and limitations of our method in Section 7. Finally, Section 8 concludes our work.

2 Related Work

2.1 Model Overfitting

Several strategies have been proposed in the literature to reduce the effects of overfitting, which can be broadly categorized as follows.

Early Stopping. This strategy aims to mitigate the phenomenon known as "learning speed slow-down." This issue occurs when the accuracy of a model ceases to improve or even worsens due to noise-learning. The concept has a long history, dating back to the 1970s in the context of the Landweber iteration [28]. It has since been widely adopted in iterative algorithms, particularly for training deep neural networks in combination with backpropagation [2].

Data Augmentation. The more complex the model, the higher the number of parameters that need to be learned. Therefore, the size of the training set must be adequate to avoid overfitting. Data augmentation techniques play a crucial role in enhancing model generalization across various domains, including pattern recognition and image processing. These techniques aim to expand existing datasets to generate additional data. Typically, four main approaches are employed [18, 32, 39]: *(i)* acquiring new data, *(ii)* introducing random noise to the existing dataset, *(iii)* reprocessing existing data to produce new instances, and *(iv)* sampling new data based on the distribution of the existing dataset.

Regularization. An overfitting model tends to incorporate all available features into its decision-making process, even those with minimal impact or that are simply noise. To address this issue, there are two main approaches: *(i) Feature selection*, where only the most relevant features are retained, discarding those deemed irrelevant; *(ii) Feature regularization*, which involves minimizing the influence of less important features by reducing their weights in the model. However, since identifying useless features can be challenging, regularization techniques work by applying a penalty term, or "regularizer", to the objective function. This penalizes complex models, encouraging simpler and more generalizable solutions. For example, L1 ("Lasso") and L2 ("Ridge") regularization, which add the L_1-norm and L_2-norm of the learned parameter as a penalties, respectively, are commonly adopted in linear regression. Furthermore, *Dropout* is a popular technique to combat overfitting in neural networks. This approach randomly drops units and relevant connections from the neural network during training to prevent co-adaptation [36].

Unlike existing regularization techniques, which either aim to reduce the magnitude of the learned model's weights (such as Lasso and Ridge) or randomly deactivate certain parameters (such as Dropout), our penalty term is inherently *data-driven*. In other words, the counterfactual regularization we introduce (CF-Reg) is computed directly from the training observations, making it highly tailored to the specific dataset under consideration.

Adversarial Training. Adversarial training is another strategy that can be adapted to mitigate overfitting. Notably, Madry et al. [25] proposed a min-max optimization framework that enhances model robustness by training against adversarial examples generated via Projected Gradient Descent (PGD). This approach serves as an effective form of regularization, as it discourages the learned model from depending on non-robust or spurious patterns [31].

2.2 Counterfactual Examples

Counterfactual examples have gained significant attention in machine learning due to their utility in various applications, including model *explainability* and *robustness*.

Explainability. Counterfactual explanations offer valuable insights into model predictions by providing alternative scenarios under which the prediction would change [35]. Several studies have explored the use of counterfactual examples to enhance the interpretability of complex machine learning models, such as ensembles of decision trees [23, 33, 34] and deep neural networks (DNNs) [21], including graph neural networks (GNNs) [3, 24]. Indeed, counterfactual instances help elucidate the underlying decision-making process of black-box models, thereby increasing trust and transparency in AI systems [4, 5, 10, 17, 26], especially when translated into natural language narratives [9].

Robustness. Counterfactual examples have also been leveraged to enhance the robustness of machine learning models against adversarial attacks [1]. Indeed, there is a strict relationship between adversarial and counterfactual examples [7], although their primary goals are divergent. While both adversarial and counterfactual examples involve perturbing input data to influence model predictions, adversarial examples are crafted to jeopardize the model, whereas counterfactual examples are generated to understand the model's behavior. By generating instances that are semantically similar to the original input but induce different model predictions, He et al. [13] enhances model robustness against adversarial perturbations.

Recent work has explored the use of generative models to produce realistic counterfactual instances for data augmentation and model refinement [8, 14], highlighting the potential of counterfactual reasoning in generative modeling tasks [22].

To the best of our knowledge, however, this is the first study to link counterfactual examples to model generalization and to employ them as the cornerstone of a new regularization strategy.

3 Background and Preliminaries

Let $f_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{Y}$ denote a predictive model parameterized by a set of (learnable) weights $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ that takes an input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and maps it to an output $y \in \mathcal{Y}$. In the standard supervised learning setting, the optimal weights ($\boldsymbol{\theta}^*$) are found by minimizing a specific loss function ($\mathcal{L}$) computed on a training set ($\mathcal{D}$) of m i.i.d. labeled instances, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Big\{ \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \Big\} = \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{|\mathcal{D}|} \sum_{i=1}^{m} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) \right\}. \tag{1}$$

Here, ℓ represents an instance-level error between the model's prediction for a given input ($\boldsymbol{x}_i$) and its corresponding actual label (y_i), such as cross-entropy (for classification tasks) or mean squared error (for regression tasks).

Furthermore, we assume to have available a counterfactual generator model $g_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{X}$, for the predictive model $f_{\boldsymbol{\theta}}$, that takes as input a data point $\boldsymbol{x}$ and produces as output its corresponding (optimal) counterfactual $\widetilde{\boldsymbol{x}}^*$. This typically requires solving a constrained objective as follows:

$$\begin{aligned} g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \widetilde{\boldsymbol{x}}^* = \arg\min_{\widetilde{\boldsymbol{x}} \in \mathcal{X}} \ & \delta(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \\ \text{s.t.: } & f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}). \end{aligned} \tag{2}$$

Here, $\delta : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{\geq 0}$ is a function that measures the distance between the original input data point and its counterfactual. In practice, δ often computes the L_1- or L_2-norm of the displacement vector between the original and the counterfactual example.

In general, for a given input $\boldsymbol{x}$, there could be several, possibly infinitely many *valid* counterfactuals: $\widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_2, \ldots$ In this work, a valid counterfactual is considered as any instance $\widetilde{\boldsymbol{x}}$ that crosses the decision boundary induced by $f_{\boldsymbol{\theta}}$, i.e., where $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}})$. More refined notions of validity are also conceivable, such as those entailing the *plausibility* of the generated counterfactual, which assesses whether the counterfactual example is indeed realistic. For example, suppose we have trained an image classifier to distinguish between birds and rabbits. In this context, a plausible (and valid) counterfactual for a rabbit must be an image resembling a bird. However, for the purpose of this work, we are only interested in characterizing the ability to find a valid counterfactual and relating this to the degree of model overfitting, regardless of its plausibility.

Formally, let $\boldsymbol{x} \in \mathcal{X}$ be an input sample, $f_{\boldsymbol{\theta}}$ a trained model, $\delta : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{\geq 0}$ a distance function, and $\varepsilon \in \mathbb{R}_{>0}$ a fixed threshold. We consider an *ε-valid counterfactual example* (ε-VCE) for $\boldsymbol{x}$ any $\widetilde{\boldsymbol{x}} \neq \boldsymbol{x}$, such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}})$ and $\delta(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \leq \varepsilon$.

4 Impact of Counterfactual Examples on Model Generalizability

4.1 Intuitive Perspective

We consider a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ of m i.i.d. labeled instances, as introduced above. Then, suppose this dataset is used to train a sequence of k different models $(f_{\boldsymbol{\theta}_0}, f_{\boldsymbol{\theta}_1}, \ldots, f_{\boldsymbol{\theta}_{k-1}})$. Each model $f_{\boldsymbol{\theta}_i}$ is associated with a training accuracy α_i (calculated on $\mathcal{D}$), such that $\alpha_0 < \alpha_1 < \ldots < \alpha_{k-1}$, where $f_{\boldsymbol{\theta}_0}$ indicates the random baseline model and $f_{\boldsymbol{\theta}_{k-1}}$ is the dummy model that simply memorizes the entire training set and, therefore, achieves the perfect training accuracy ($\alpha_{k-1} = 1$).[1]

We claim that for a fixed positive threshold $\varepsilon > 0$, the expected fraction of training points for which we can find an ε-valid counterfactual example is positively correlated with the training accuracy of the model. In other words, given two trained models $f_{\boldsymbol{\theta}_i}$ and $f_{\boldsymbol{\theta}_j}$, where $i > j$, whose training accuracies are $\alpha_i > \alpha_j$, we expect to find, on average, more ε-valid counterfactual examples for $f_{\boldsymbol{\theta}_i}$.

This intuition stems from the idea that a model with a higher training accuracy tends to have a more intricate decision boundary surface, which captures all the nuances of the training data more precisely. Consequently, data points are, *on average*, closer to such a convoluted decision boundary, making it easier to cross it and thus to find valid counterfactual examples within a distance of ε.

In essence, a model for which finding counterfactual examples is "too easy" may indicate a risk of overfitting. Thus, a trade-off must exist between the model's generalizability and its counterfactual explainability, which we aim to investigate further in this study.

4.2 Theoretical Motivation

The intuition outlined above is supported by a well-established theoretical foundation rooted in classical margin theory. Statistical learning theory tells us that a model's generalization error can be bounded using empirical margin distributions [19]. A prominent example is the support vector machine (SVM), which *explicitly* maximizes the minimum margin on linearly separable datasets.

Interestingly, margin maximization is not exclusive to SVMs. Soudry et al. [29] show that gradient descent applied to logistic regression on linearly separable data *implicitly* maximizes the minimum margin, thereby mirroring the behavior of SVMs. This implicit bias toward margin maximization has also been observed in more complex models, including deep neural networks [11, 15].

Building on these insights, Wu and Yu [38] uncover an intriguing trade-off between the minimum and *average* margins, where the latter is defined as the average distance of training examples to the decision boundary. Specifically, they show that optimizing for a larger minimum margin can unintentionally reduce the average margin, thereby increasing the model's vulnerability to adversarial examples. For the same reason, as training progresses and the average distance between data points and the decision boundary decreases, it becomes easier to identify a valid counterfactual example within a fixed perturbation radius ε.

To further validate this claim, Figure 2 shows the empirical distribution of margin distances computed from all training points in the `Water` dataset [16] at different training epochs of a logistic regression model. We notice that, at initialization (epoch 0), the distribution is relatively uniform, with a higher average margin distance of 0.754. As training progresses, the distribution of distances to the classifier's boundary becomes increasingly right-skewed, concentrating more of the mass near lower margin values. By epoch 8,000, the average margin distance drops to 0.202.

[1] In practice, the sequence of k models may correspond to intermediate checkpoints saved at various training epochs.

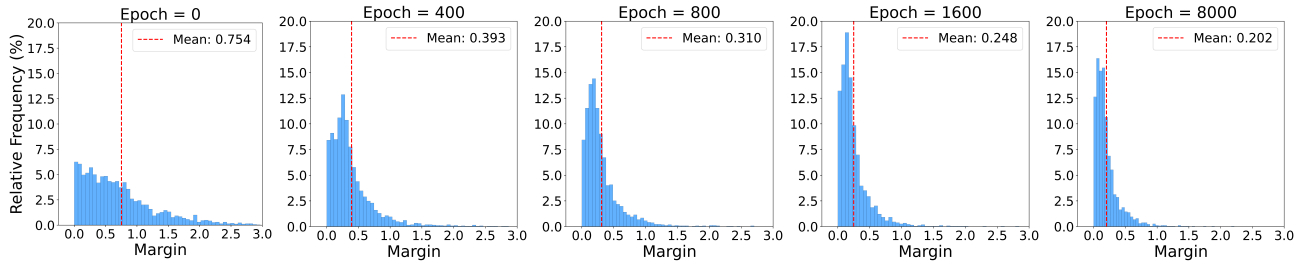


Figure 2: Evolution of the empirical distribution of margin distances for training data points in the `Water` dataset across different training epochs of a logistic regression model. As the training progresses, the *average* margin distance decreases.

4.3 Estimating Counterfactual Probability

In this section, we formally characterize what we mean by the ease of finding an ε-valid counterfactual example for a given data point and, therefore, for a full training set of data points.

Let us consider the generic predictive model $f_{\boldsymbol{\theta}}$ trained on $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$. Suppose we associate with each training data point $\boldsymbol{x}_i$ a binary random variable $X_i \in \{0, 1\}$, which indicates whether there exists an ε-valid counterfactual example $\widetilde{\boldsymbol{x}}_i$ for $\boldsymbol{x}_i$. Therefore, each X_i follows a Bernoulli distribution, i.e., $X_i \sim \text{Bernoulli}(p_i^\varepsilon)$, whose probability mass function is defined as below.

$$\mathbb{P}(X_i = k) = p_{X_i}(k; p_i^\varepsilon) = \begin{cases} p_i^\varepsilon & \text{, if } k = 1 \\ 1 - p_i^\varepsilon & \text{, if } k = 0. \end{cases} \tag{3}$$

Furthermore, let $\bar{X}$ denote the fraction of training points for which an ε-valid counterfactual example exists. Formally, $\bar{X} = \frac{1}{m}\sum_{i=1}^m X_i$ is the average of m Bernoulli random variables.

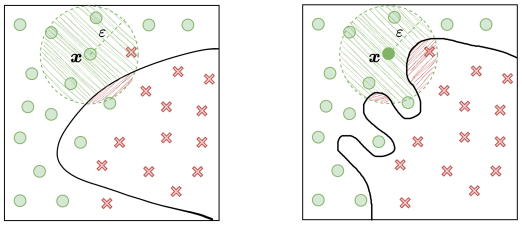
By the linearity of expectation, we can calculate:

$$\mathbb{E}[\bar{X}] = \mathbb{E}\Big[\frac{1}{m}\sum_{i=1}^m X_i\Big] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}[X_i], \tag{4}$$

where $\mathbb{E}[X_i] = p_i^\varepsilon$ and, therefore, $\mathbb{E}[\bar{X}] = \bar{p}^\varepsilon$ is the average across the entire dataset. Note that the random variable $Z = \sum_{i=1}^m X_i$ follows a Poisson-Binomial distribution, since X_i's are independent but not necessarily identically distributed. Consequently, $\bar{X} = \frac{Z}{m}$ is also a Poisson-Binomial random variable, scaled by a factor of $1/m$.

For a fixed $\varepsilon > 0$, we argue that $\mathbb{E}[\bar{X}]$ is positively correlated with the model's training accuracy. This follows from the theoretical result discussed above [38], which shows that higher training accuracy corresponds to a smaller average margin to the decision boundary. Therefore, as the training accuracy of $f_{\boldsymbol{\theta}}$ increases, so does the average fraction of training instances for which a valid counterfactual can be found within distance ε.

Firstly, we need to estimate each p_i^ε, which we refer to as the (sample-level) *ε-valid counterfactual probability* (ε-VCP). Formally, let $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ be a generic training point, and $\varepsilon \in \mathbb{R}_{>0}$ a positive real number. Consider the ε n-ball as the n-dimensional hypersphere of radius ε centered around $\boldsymbol{x}$. Let $V_{\boldsymbol{x}}^\varepsilon \subseteq \mathbb{R}^n$ be the total hypervolume of this hypersphere, and let $V_{\widetilde{\boldsymbol{x}}}^\varepsilon \subseteq V_{\boldsymbol{x}}^\varepsilon$ denote the portion of the total hypervolume falling within the counterfactual region delimited by the decision boundary induced by $f_{\boldsymbol{\theta}}$. Therefore, we can estimate the ε-VCP for $\boldsymbol{x}$, as $p^\varepsilon = V_{\widetilde{\boldsymbol{x}}}^\varepsilon / V_{\boldsymbol{x}}^\varepsilon$.



(a) Low ε-valid counterfactual probability

(b) High ε-valid counterfactual probability

Figure 3: The *ε-valid counterfactual probability* for a sample $\boldsymbol{x} \in \mathbb{R}^2$ can be estimated as the ratio of the area of the circle centered in $\boldsymbol{x}$ with radius ε that falls behind the decision boundary (in red).

Intuitively, given a fixed $\boldsymbol{x}$ and two models having different decision boundaries, it will generally be much easier to find an ε-valid counterfactual if the hypervolume $V_{\widetilde{\boldsymbol{x}}}^\varepsilon$ is larger. This intuition is depicted in Figure 3, which illustrates how the probability of finding an ε-valid counterfactual for a 2-dimensional data point may increase with model overfitting.

To estimate $V_{\widetilde{\boldsymbol{x}}}^\varepsilon$, we can apply standard Monte Carlo integration, a well-established technique that uses random draws to numerically compute a definite integral. Furthermore, using Monte Carlo integration will provide valuable knowledge on the shape of a model's decision boundary as a by-product. The estimate outlined above assumes that data points are uniformly distributed around $\boldsymbol{x}$ in the input space, which may be reasonable for appropriate values of ε. Indeed, it is worth noticing that $\boldsymbol{x}$ can be an embedding from an autoencoding process, mapping each raw training point into a dense, lower-dimensional latent manifold within the original, high-dimensional space. More accurate estimates could be obtained by sampling the ε-neighborhood from the manifold using advanced strategies (e.g., see Chen et al. [5]). However, this lies beyond the primary scope of this step and will be explored in future work.

Finally, if we extend the reasoning above to *all* training points, we should observe that $\mathbb{E}[\bar{X}] = \bar{p}^\varepsilon$ is higher for overtrained models.

4.4 Empirical Validation

To investigate the impact of overfitting on ε-VCP, we trained two Multi-Layer Perceptron (MLP) models on the `Water` dataset [16] for a binary classification task. Both models used the same 5-layer

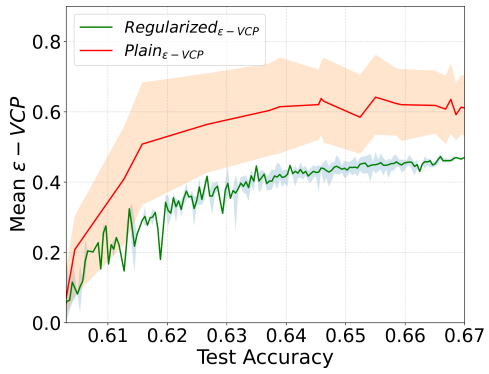


Figure 4: The mean ε-VCP (y-axis) vs. the model's training accuracy (x-axis). *Plain*$_{\varepsilon\text{-VCP}}$ is the "vanilla" MLP, while *Regularized*$_{\varepsilon\text{-VCP}}$ is the same MLP yet with a dropout rate of 0.5.

architecture, but only one employed dropout regularization with a rate of 0.5. The models were trained over 500 epochs using the Adam optimizer with a learning rate of $\eta = 0.001$. The ε-VCP was estimated through Monte Carlo integration, as discussed in Section 4.3. Specifically, it was computed as $p^{\varepsilon} = V^{\varepsilon}_{\widetilde{\boldsymbol{x}}}/V^{\varepsilon}_{\boldsymbol{x}}$, where $V^{\varepsilon}_{\widetilde{\boldsymbol{x}}}$ was approximated using 100 random samples per training point $\boldsymbol{x}$.

It is important to note that the choice of ε is inherently dependent on the characteristics of the dataset and is best determined empirically to ensure a meaningful evaluation of counterfactual validity while preserving consistency with the data distribution. As ε defines the maximum norm of the perturbation vector applied to each instance, it delineates the feasible search space for generating counterfactuals. An overly small value may excessively constrain the perturbations, limiting the ability to capture substantive changes in model predictions. In contrast, a value that is too large can produce counterfactuals that are implausible or inconsistent with the underlying data structure. In this experiment with the `Water` dataset, we found that setting $\varepsilon = 1.5$ provides a suitable trade-off, allowing us to capture a diverse yet interpretable range of counterfactual examples without introducing excessive modifications to the original training points.

Figure 4 illustrates the evolution of the average ε-VCP, calculated over all training points, as a function of the models' training accuracy. From this plot, three key insights emerge. First, the average ε-VCP increases alongside training accuracy for both models, confirming our hypothesis that the likelihood of finding valid counterfactuals rises as models tend to overfit. Second, the plain, unregularized MLP exhibits higher ε-VCP values, suggesting that its more complex decision boundary makes it easier to identify valid counterfactual examples. Third, the trend persists while the MLP trained with dropout regularization exhibits a less pronounced increase (i.e., smaller ε-VCP values). This suggests that although dropout regularization smooths the decision boundary to some extent, it may not sufficiently counteract this phenomenon, leaving room for further improvement.

5 Counterfactual Regularization (CF-Reg)

5.1 A New Training Loss

We exploit the correlation between overfitting and the ability to find counterfactual examples, as highlighted in the previous section, to define a new regularized training loss as follows.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Big\{ \mathcal{L}_{\text{emp}}(\boldsymbol{\theta}; \mathcal{D}) - \alpha \mathcal{L}_{\text{cf}}(\boldsymbol{\theta}; \mathcal{D}, \varepsilon) \Big\}. \tag{5}$$

The first term ($\mathcal{L}_{\text{emp}}$) is the standard empirical risk, while the second term ($\mathcal{L}_{\text{cf}}$) is our proposed *counterfactual regularization* component, with $\alpha \in \mathbb{R}_{\geq 0}$ serving as a hyperparameter to weigh its contribution. More specifically, the counterfactual regularization component can be defined as follows:

$$\mathcal{L}_{\text{cf}}(\boldsymbol{\theta}; \mathcal{D}, \varepsilon) = \varphi(\{d(\boldsymbol{x}_i, g_{\boldsymbol{\theta}}(\boldsymbol{x}_i)),\ \boldsymbol{x}_i \in \mathcal{D}\}; \varepsilon), \tag{6}$$

where $\varphi(\cdot; \varepsilon)$ – parameterized by ε – acts as an aggregation function applied to the set of distances $d(\boldsymbol{x}_i, g_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ between each sample $\boldsymbol{x}_i$ and its corresponding counterfactual generated by the model, $g_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. Intuitively, the closer an instance is to its counterfactual, the greater the penalty imposed by the newly introduced loss. In other words, optimizing the objective defined in (5) aims to ensure a sufficient margin between each training point and its corresponding counterfactual. Moreover, we would like this penalty to be stronger for training points where finding a counterfactual is easier – i.e., those closer to the decision boundary. To achieve this goal, each distance $d(\boldsymbol{x}_i, g_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ can be assigned a weight w_i^{ε} that depends on ε during aggregation. For example, this weight may be set as the ε-VCP associated with each training instance $\boldsymbol{x}_i$ (i.e., $w_i^{\varepsilon} = p_i^{\varepsilon}$).

Note that, for each data point $\boldsymbol{x}_i$, its corresponding p_i^{ε} can be estimated as described in Section 4.3. However, since p_i^{ε} depends on the shape of the decision boundary, which may evolve dynamically across training epochs, its estimation should ideally be updated at *every* epoch. Therefore, this could introduce a significant computational overhead in the training process, which we can mitigate by updating our estimates $\{p_i^{\varepsilon}\}_{i=1}^{m}$ periodically, for example, every t epochs. We conjecture that a trade-off exists between the effectiveness of the counterfactual regularization term – impacted by the accuracy of each estimate p_i^{ε} – and the computational cost of the training process.

It is worth noticing that our counterfactual regularizer – hereinafter referred to as CF-Reg – is flexible enough to support any choice of aggregation function (φ), distance (d), and counterfactual generator ($g_{\boldsymbol{\theta}}$). However, the optimization problem in (5) can be efficiently solved using standard gradient-based methods, provided that φ, d, and $g_{\boldsymbol{\theta}}$ are all differentiable with respect to $\boldsymbol{\theta}$.

In this work, we set φ as the mean and d as the Euclidean distance (i.e., the L_2-norm of the displacement vector resulting from the difference between the original sample and its counterfactual). For $g_{\boldsymbol{\theta}}$, we adopt the *score counterfactual explanation* method proposed by Wachter et al. [35], and we discuss the rationale behind this choice below. Alternative formulations of these components are possible and will be explored in future studies.

5.2 Counterfactual Example Generator

A key component of CF-Reg is the counterfactual generator $g_{\boldsymbol{\theta}}$, as this is used to compute the optimal counterfactual example for each training data point. To incorporate this step into the regularized

training process, the counterfactual generator must satisfy two essential requirements. First, it must be differentiable with respect to the predictive model weights $\boldsymbol{\theta}$. Second, it must be highly efficient to ensure the overall feasibility of our method.

Among the various counterfactual generation approaches in the literature, we selected the *score counterfactual explanation* method proposed by [35], which operates as follows. Let $\boldsymbol{x} \in \mathcal{X}$ be an input sample, $f_{\boldsymbol{\theta}}$ a trained model, $\delta : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{\geq 0}$ a distance function, $\beta \in \mathbb{R}_{\geq 0}$ a controlling hyperparameter, and $s \in \mathbb{R}$ a target score. Thus, the score counterfactual explanation $\widetilde{\boldsymbol{x}}$ for $\boldsymbol{x}$ is the result of the following optimization problem:

$$\widetilde{\boldsymbol{x}}^* = \arg\min_{\widetilde{\boldsymbol{x}} \in \mathcal{X}} \left(f_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}) - s\right)^2 + \beta\delta(\boldsymbol{x}, \widetilde{\boldsymbol{x}}). \tag{7}$$

With a slight abuse of notation, in (7), we treat the output of the predictive model $f_{\boldsymbol{\theta}}$ as a continuous value, even for classification tasks. To accommodate this requirement, we can, for instance, assume access to the logits, which are then passed through the appropriate activation function, such as sigmoid or softmax.

Note that the resulting counterfactual generator satisfies the two requirements mentioned above. In particular, by carefully selecting the distance function δ in (7), the resulting objective becomes differentiable with respect to the predictive model weights $\boldsymbol{\theta}$. Furthermore, when using the score-based counterfactual explanation approach, determining the optimal counterfactual perturbation vector $\boldsymbol{\delta} = \boldsymbol{x} - \widetilde{\boldsymbol{x}}^* = \boldsymbol{x} - g_{\boldsymbol{\theta}}(\boldsymbol{x})$ – i.e., the displacement vector between the original instance and its optimal counterfactual – admits a closed-form solution under specific assumptions. Specifically, if $f_{\boldsymbol{\theta}}$ is a linear function and $\widetilde{\boldsymbol{x}}^* = g_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the optimal counterfactual example for the input $\boldsymbol{x}$, Pawelczyk et al. [27] proved that:

$$\boldsymbol{\delta} = \frac{t}{\beta + \|\boldsymbol{\theta}\|^2} \cdot \boldsymbol{\theta}, \tag{8}$$

where $t = s - f_{\boldsymbol{\theta}}(\boldsymbol{x})$. This makes the method highly efficient and well-suited for incorporation into our regularized training loss.

From (8), we can, therefore, compute the L_2-norm of the perturbation vector $\boldsymbol{\delta}_i$, for each training instance $\boldsymbol{x}_i$. This value is then used as $d(\boldsymbol{x}_i, \widetilde{\boldsymbol{x}}_i^*)$, as required by the proposed regularized training loss in (6). Finally, the L_2-norms of all perturbation vectors are aggregated using the function $\varphi(\cdot; \varepsilon)$, resulting in the following (weighted) average:

$$\overline{||\boldsymbol{\delta}||} = \frac{1}{m} \sum_{i=1}^{m} w_i^{\varepsilon} * ||\boldsymbol{\delta}_i||. \tag{9}$$

In this work, weights are uniformly set as $w_i^{\varepsilon} = 1 \; \forall i \in \{1, \ldots, m\}$, thus reducing the aggregation to a plain average. Alternatively, more sophisticated strategies can be used, such as the one outlined in Section 5.1, where each training point is assigned a weight corresponding to its estimated ε-VCP, i.e., $w_i^{\varepsilon} = p_i^{\varepsilon}$.

Overall, this transforms our counterfactual regularized loss into:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}_{\text{emp}}(\boldsymbol{\theta}; \mathcal{D}) - \alpha \overline{||\boldsymbol{\delta}||} \right\}. \tag{10}$$

Finally, the objective in (10) can be solved using standard gradient-based optimization methods.

6 Experiments

In this section, we evaluate the ability of the proposed counterfactual regularized loss (CF-Reg), as defined in (10), to mitigate overfitting when compared to existing regularization techniques. In Appendix B, instead, we analyze the relationship between the test loss and the counterfactual vector norm during training.

6.1 Experimental Setup

Datasets, Models, and Tasks. The experiments are run on four datasets: `Water` [16], `Phoneme` [6], `Higgs` [37], and `CIFAR-10` [20]. Furthermore, we consider the following models: Logistic Regression (LR), two Multi-Layer Perceptrons ($\text{MLP}_{\text{small}}$ and $\text{MLP}_{\text{large}}$), and *PreactResNet-18* [12] as a representative Convolutional Neural Network (CNN) architecture. We focus on binary classification tasks and leave the extension to multiclass scenarios for future work. However, for datasets originally designed for multiclass classification, we adapt them by selecting two classes to align with our binary setting. Additional details are provided in Appendix A.1.

Evaluation Measures. To characterize the degree of overfitting, we use the test loss, as it serves as a reliable indicator of the model's generalization capability to unseen data. Additionally, we evaluate the predictive performance of each model using the test accuracy.

Baselines. We compare CF-Reg with the following methods: Early Stopping, Dropout, adversarial training via well-known projected gradient descent (PGD) [25], two regularizers – i.e., L1-Reg ("Lasso") and L2-Reg ("Ridge") – and no regularization (No-Reg).

Hyperparameter Settings. We analyze the sensitivity of CF-Reg to its hyperparameters in Section 6.3. Due to space constraints, a comprehensive discussion covering all regularization methods, including the tuning strategies adopted, is provided in Appendix C. The best selected values are summarized in Table 7.

Implementation Details. To deliberately induce overfitting in logistic regression models, we apply a polynomial feature expansion that increases the input dimensionality beyond the number of training samples. This ensures that the model has sufficient capacity to memorize the training data, thereby allowing us to assess the effect of our counterfactual regularizer. The degree of the polynomial expansion is selected as the smallest value that results in a number of features exceeding the number of training instances. Note that this preprocessing step is not required for neural network models, as they are chosen to be complex enough to overfit the data.

In addition, to leverage the closed-form solution for computing the optimal perturbation vector as defined in (8), we use a local linear approximation of the model during the counterfactual generation process. Thus, given an instance $\boldsymbol{x}_i$, we compute the (optimal) counterfactual not with respect to the original model $f_{\boldsymbol{\theta}}$, but with respect to its first-order Taylor expansion around $\boldsymbol{x}_i$, denoted as:

$$f_{\boldsymbol{\theta}}^{lin}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)(\boldsymbol{x} - \boldsymbol{x}_i). \tag{11}$$

Note that this step is unnecessary for logistic regression, as it is inherently a linear model.

We run all experiments on a machine equipped with an AMD Ryzen 9 7900 12-Core Processor and an NVIDIA GeForce RTX 4090 GPU. Our implementation is based on the PyTorch Lightning framework, and the source code for our experiments is available at

the following GitHub repository: https://github.com/hercolelab/CF-Reg. Further details are described in Appendix A.2.

6.2 Effectiveness of CF-Reg

We compare the performance of CF-Reg against the baselines indicated in Section 6.1. For each model and dataset combination, Table 1 shows the mean value and standard deviation of test accuracy achieved by each method across 5 random initializations. From the results in Table 1, the following key insights can be drawn.

General Validity of CF-Reg. Across the board, CF-Reg demonstrates strong performance, particularly on tabular datasets such as `Water`, `Phoneme`, and `Higgs`. In many configurations, it either matches or outperforms all other baselines, and in several cases, the improvements are statistically significant. This supports the core hypothesis behind CF-Reg: that training with informative counterfactual examples can improve generalization by encouraging decision boundary robustness in data-relevant directions.

Comparison with other Regularization Techniques. CF-Reg consistently outperforms traditional penalty-based regularizers like L1-Reg and L2-Reg. These methods penalize model complexity indirectly through weight constraints, whereas CF-Reg directly encourages decision-level stability, which appears more effective in preserving model generalization. Moreover, CF-Reg performs on par with or better than implicit regularization methods such as early stopping and dropout. Notably, in cases where CF-Reg underperforms slightly, it still retains a critical advantage: it leverages the full expressive capacity of the architecture, rather than curtailing it through early termination of training or by randomly omitting connections during forward passes. This is particularly relevant for modern architectures where expressiveness is key to capturing subtle data patterns. When compared to adversarial training (PGD), CF-Reg also shows almost always superior or comparable accuracy.

Handling Complex Input Domains. On the `CIFAR-10` dataset, CF-Reg does not outperform traditional regularizers. In fact, in this context, the No-Reg baseline performs the best, suggesting that overfitting is less of a concern, perhaps due to the indirect regularizing effect of convolutional architectures. Additionally, the current counterfactual generation strategy appears less effective for image data, where generating meaningful perturbations is substantially harder. This points to a promising direction for future work: using more expressive and domain-aware counterfactual generators tailored to high-dimensional, unstructured inputs, in order to fully extend the applicability of CF-Reg to complex data.

6.3 Hyperparameter Sensitivity Analysis

CF-Reg relies on two key hyperparameters: α and β. The former is intrinsic to the loss formulation defined in (2), while the latter is closely tied to the choice of the score-based counterfactual explanation method used.

Figure 5 illustrates how the test accuracy of an MLP trained on the `Water` dataset changes for different combinations of α and β.

We observe that, for a fixed β, increasing the weight of our counterfactual regularizer (α) can slightly improve test accuracy until a sharp drop occurs for $\alpha > 0.1$. This behavior is expected, as the impact of CF-Reg, like any regularization term, can be disruptive

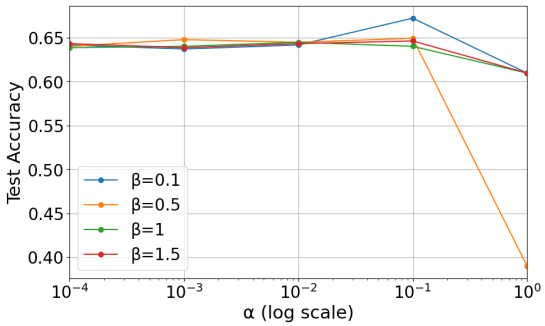


Figure 5: The test accuracy of an MLP trained on the `Water` dataset, evaluated while varying the weight of our counterfactual regularizer (α) for different values of β.

if not properly calibrated. Moreover, this finding confirms that CF-Reg is inherently data-driven and, thus, requires specific fine-tuning based on the combination of the model and dataset at hand.

The best values for α and β are shown in Table 7 (Appendix C).

6.4 Comparison of Weighting Schemes

In our current implementation, we assign all training instances equal importance ($w_i^{\varepsilon} = 1, \forall i$). While simple, this uniform weighting may limit the full potential of CF-Reg. Adopting a more fine-grained weighting strategy could better balance the regularization strength across samples, albeit at the cost of increased training complexity. To this end, in this section, we present the results of a comparison among three different weighting schemes. The first scheme is the uniform weighting adopted to produce the results in Table 1. The second scheme weights each data point according to its ε-VCP value. The third scheme, denoted as TPTN, assigns zero weight to misclassified samples in the regularization term.

The results reported in Table 3 indicate that, within our small-scale experimental setting, more complex weighting strategies do not yield a clear advantage over simple uniform weighting.

Table 3: Mean and standard deviation of test accuracy over 5 random initializations for each combination of model, dataset, and weighting scheme.

Model	Dataset	Uniform	ε-VCP	TPTN
LR	`Water`	0.6915 ± 0.0017	0.6345 ± 0.0091	0.6902 ± 0.0029
$\text{MLP}_{\text{small}}$	`Water`	0.6796 ± 0.0045	0.6195 ± 0.0219	0.6716 ± 0.0154
$\text{MLP}_{\text{large}}$	`Water`	0.6787 ± 0.0092	0.6110 ± 0.0234	0.6601 ± 0.0248
LR	`Phoneme`	0.8764 ± 0.0084	0.8792 ± 0.0059	0.7874 ± 0.0048
$\text{MLP}_{\text{small}}$	`Phoneme`	0.9005 ± 0.0065	0.9031 ± 0.0045	0.8982 ± 0.0073
$\text{MLP}_{\text{large}}$	`Phoneme`	0.9149 ± 0.0024	0.9016 ± 0.0091	0.9062 ± 0.0055
$\text{MLP}_{\text{small}}$	`Higgs`	0.7245 ± 0.0057	0.7258 ± 0.0036	0.7248 ± 0.0040
$\text{MLP}_{\text{large}}$	`Higgs`	0.7278 ± 0.0027	0.6931 ± 0.0041	0.7272 ± 0.0031

7 Discussion

7.1 Feasibility of CF-Reg

As with any regularizer, CF-Reg introduces some computational overhead in exchange for improved model generalization. Specifically, the extra cost stems from the need to compute a counterfactual

Table 1: Mean and standard deviation of test accuracy over 5 random initializations for each combination of model, dataset, and regularization method. Best results are shown in bold; results that are statistically significantly better than the second-best (at the $\alpha = 0.05$ significance level) are marked with a *.

Model	Dataset	No-Reg	L1-Reg	L2-Reg	Dropout	Early Stopping	PGD	CF-Reg (ours)
LR	Water	0.6030 ± 0.0053	0.6652 ± 0.0053	0.6677 ± 0.0114	N/A	0.6098 ± 0.0000	0.6384 ± 0.0183	**0.6915 ± 0.0017***
$\text{MLP}_{\text{small}}$	Water	0.6128 ± 0.0103	0.6098 ± 0.0000	0.6759 ± 0.0065	0.6515 ± 0.0100	0.6765 ± 0.0059	0.6524 ± 0.0011	**0.6796 ± 0.0045**
$\text{MLP}_{\text{large}}$	Water	0.6168 ± 0.0063	0.6098 ± 0.0000	0.6320 ± 0.0158	0.6564 ± 0.0104	0.6573 ± 0.0112	0.6464 ± 0.0053	**0.6787 ± 0.0092***
LR	Phoneme	0.8729 ± 0.0052	0.8157 ± 0.0036	0.8427 ± 0.0078	N/A	0.8622 ± 0.0109	0.7353 ± 0.0006	**0.8764 ± 0.0084**
$\text{MLP}_{\text{small}}$	Phoneme	**0.9016 ± 0.0088**	0.8511 ± 0.0092	0.7963 ± 0.0041	**0.9016 ± 0.0059**	0.8957 ± 0.0105	0.8915 ± 0.0029	0.9005 ± 0.0065
$\text{MLP}_{\text{large}}$	Phoneme	0.9101 ± 0.0026	0.7068 ± 0.0000	0.8735 ± 0.0042	0.9099 ± 0.0078	0.9066 ± 0.0083	0.9021 ± 0.0021	**0.9149 ± 0.0024***
$\text{MLP}_{\text{small}}$	Higgs	0.7204 ± 0.0043	0.4706 ± 0.0000	0.7231 ± 0.0033	0.7210 ± 0.0016	0.7271 ± 0.0049	**0.7334 ± 0.0025***	0.7245 ± 0.0057
$\text{MLP}_{\text{large}}$	Higgs	0.6904 ± 0.0020	0.4706 ± 0.0000	0.7223 ± 0.0056	0.7323 ± 0.0039	0.7149 ± 0.0043	0.6995 ± 0.0018	**0.7278 ± 0.0027**
CNN	CIFAR-10	**0.8415 ± 0.0059**	0.8355 ± 0.0160	0.8408 ± 0.0035	N/A	0.8413 ± 0.0008	0.7071 ± 0.0316	0.8388 ± 0.0106

Table 2: Mean and standard deviation of training time (in seconds) over 5 independent runs. Each value refers to the training process for the corresponding entry in Table 1.

Model	Dataset	No-Reg	L1-Reg	L2-Reg	Dropout	Early Stopping	PGD	CF-Reg (ours)
LR	Water	19.86 ± 0.51	20.87 ± 0.33	21.06 ± 0.29	N/A	0.46 ± 0.11	119.23 ± 0.83	21.78 ± 0.54
$\text{MLP}_{\text{small}}$	Water	19.78 ± 0.21	22.47 ± 0.47	22.60 ± 0.41	21.20 ± 0.26	1.37 ± 0.49	120.31 ± 0.05	23.07 ± 0.28
$\text{MLP}_{\text{large}}$	Water	22.94 ± 0.37	26.25 ± 0.18	27.21 ± 0.42	25.88 ± 0.45	0.76 ± 0.10	157.83 ± 0.05	27.48 ± 0.29
LR	Phoneme	29.54 ± 0.40	32.25 ± 0.68	32.22 ± 0.23	N/A	20.40 ± 3.27	128.67 ± 0.05	33.84 ± 0.37
$\text{MLP}_{\text{small}}$	Phoneme	31.02 ± 0.56	34.90 ± 0.35	35.55 ± 0.72	33.12 ± 0.18	31.33 ± 0.07	565.60 ± 1.67	36.60 ± 0.72
$\text{MLP}_{\text{large}}$	Phoneme	35.66 ± 0.60	41.27 ± 0.99	42.60 ± 0.59	40.25 ± 0.70	20.95 ± 0.70	253.80 ± 0.83	43.95 ± 0.56
$\text{MLP}_{\text{small}}$	Higgs	415.16 ± 4.70	470.94 ± 8.46	485.45 ± 8.22	448.73 ± 5.04	156.89 ± 7.03	459.40 ± 1.67	506.11 ± 5.73
$\text{MLP}_{\text{large}}$	Higgs	486.08 ± 5.35	571.48 ± 10.70	593.64 ± 8.38	545.84 ± 23.19	29.38 ± 2.73	584.10 ± 1.41	613.39 ± 10.71
CNN	CIFAR-10	377.47 ± 1.66	401.37 ± 1.94	404.53 ± 3.54	N/A	318.19 ± 2.20	1304.40 ± 5.62	1294.23 ± 3.88

example *for each* training instance during learning. Since CF-Reg is compatible with any counterfactual generator, the actual overhead depends on the complexity of the generator used.

Indeed, more advanced counterfactual generators may yield more effective regularization but require longer computation times. Conversely, simpler generators, such as the score-based model employed in this study, offer efficient training at the potential cost of regularization quality. We analyze this trade-off in greater depth in Section 7.3.

We quantify the cost of CF-Reg by evaluating its impact on model training time. Specifically, Table 2 reports the mean and standard deviation of training time for each model-dataset pair presented in Table 1. We observe that CF-Reg introduces only a modest computational overhead when employing the score-based generator. However, on larger datasets, the need to compute counterfactual distances for every training point leads to increased training time, as expected. Interestingly, the number of model parameters does not significantly affect the training time. Instead, input dimensionality plays a more substantial role, especially for nonlinear models. For example, in CIFAR-10, gradients for local linearization must be computed in a high-dimensional space, $\mathbb{R}^{3\times32\times32}$, which contributes considerably to the increased training cost.

7.2 Counterfactual Explanations as a By-Product

While CF-Reg introduces some computational overhead and does not always significantly outperform all baselines, it offers a unique advantage not provided by traditional regularizers: the generation of counterfactual explanations as a natural by-product of training. These explanations, typically computed through separate and often costly post hoc procedures [5, 26], are instead amortized over training with CF-Reg, yielding improved generalization, possibly without additional inference cost.

At test time, given a new input $\boldsymbol{x}_{\text{new}}$, approximate counterfactual explanations for the prediction $f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{new}})$ can be efficiently retrieved by: (i) identifying the k nearest training instances to $\boldsymbol{x}_{\text{new}}$, and (ii) returning their precomputed counterfactual examples via a simple lookup, which takes constant time ($O(1)$) per instance.

7.3 Main Limitations and Future Improvements

In this section, we discuss the main limitations of CF-Reg and outline directions for future improvements. Our method was designed to be general and efficient; however, some design choices, while practical, may limit performance in certain settings. Below, we sketch the key areas that are worth investigating.

Efficiency vs. Regularization Trade-off. As introduced in Section 5.1, our framework supports *any* counterfactual generator $g_{\boldsymbol{\theta}}$, but this flexibility introduces variability in computational cost. When using lightweight generators (e.g., score-based), training remains fast, but the regularization may be less effective. In contrast, more expressive generators could improve generalization at the expense of longer training times. Exploring this trade-off and the use of more sophisticated counterfactual generators is a promising avenue for future research.

To improve the feasibility of CF-Reg, we plan to investigate the following strategies: (i) Cache and update counterfactuals every

t epochs instead of at each iteration; (ii) Cluster training points and generate one counterfactual per cluster rather than per sample; (iii) Perform counterfactual search only for a selected subset of influential training points, identified via random sampling or heuristic criteria.

Linear Approximation. Even when using the score-based generator, we do not fully exploit the closed-form solution to compute the optimal perturbation vector in (8), as it assumes f_{θ} to be a linear model, which generally does not hold for deep neural networks. Instead, we adopt the first-order Taylor approximation in (11), which may result in suboptimal counterfactuals due to its limited ability to capture the model's nonlinear behavior.

Counterfactual Generator Learning. In this work, we treat the counterfactual generator g_{θ} as a fixed component, queried during the regularized training of the primary model f_{θ}. A promising direction for future research is to integrate the learning of g_{θ} directly into the training process, thereby formulating the task as a bilevel optimization problem. Solving this problem would allow for the joint learning of both the predictive model f_{θ} and its associated counterfactual generator g_{θ}.

8 Conclusion

We introduced and analyzed the trade-off between the generalizability and explainability of predictive models, highlighting their often conflicting nature. Specifically, based on well-known theoretical results in margin theory, we demonstrated that the degree of model overfitting is positively correlated with its ability to generate counterfactual examples. Building on this insight, we proposed CF-Reg, a novel regularization technique that integrates a counterfactual regularization term into the training objective, to balance between predictive performance and interpretability.

Through extensive experiments across multiple datasets and model architectures, we showed that CF-Reg generally outperforms existing regularization techniques, improving generalization while simultaneously generating meaningful counterfactual explanations. Our results suggest that counterfactual-based regularization can serve as a principled approach to improving model robustness without sacrificing interpretability.

Finally, we outlined promising directions for future work, primarily aimed at improving the efficiency of our method.

Acknowledgments

This work was partially supported by the following projects: FAIR (PE0000013) and SERICS (PE00000014), funded under the National Recovery and Resilience Plan by the European Union - NextGenerationEU; HypeKG – Hybrid Prediction and Explanation with Knowledge Graphs (2022Y34XNM) and NEREO – Neural Reasoning Over Open Data (2022AEFHA), funded by the Italian Ministry of University and Research under the PRIN 2022 program; GHOST – Protecting User Privacy from Community Detection in Social Networks (B83C24007070005) and SEEDS – Sustainable Ecosystems in Evolving Digital Societies, funded by Sapienza University of Rome through the "Progetti di Ricerca Grandi" and "Progetti Dipartimentali" programs, respectively.

References

[1] Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. 2018. Unrestricted Adversarial Examples. *arXiv preprint arXiv:1809.08352* (2018).

[2] Rich Caruana, Steve Lawrence, and C. Giles. 2000. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp (Eds.), Vol. 13. MIT Press, 381–387. https://proceedings.neurips.cc/paper_files/paper/2000/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf

[3] Ziheng Chen, Jin Huang, Fabrizio Silvestri, Yongfeng Zhang, Hongshik Ahn, and Gabriele Tolomei. 2025. Joint Factual and Counterfactual Explanations for Top-k GNN-based Recommendations. *ACM Trans. Recomm. Syst.* (May 2025). https://doi.org/10.1145/3731683 Just Accepted.

[4] Ziheng Chen, Fabrizio Silvestri, Gabriele Tolomei, Jia Wang, He Zhu, and Hongshik Ahn. 2024. Explain the Explainer: Interpreting Model-Agnostic Counterfactual Explanations of a Deep Reinforcement Learning Agent. *IEEE Transactions on Artificial Intelligence* 5, 4 (2024), 1443–1457. https://doi.org/10.1109/TAI.2022.3223892

[5] Ziheng Chen, Fabrizio Silvestri, Jia Wang, He Zhu, Hongshik Ahn, and Gabriele Tolomei. 2022. ReLAX: Reinforcement Learning Agent Explainer for Arbitrary Predictive Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 252–261. https://doi.org/10.1145/3511808.3557429

[6] Thomson-Sintra Dominique Van Cappel. 1993. Phoneme. https://www.openml.org/search?type=data&sort=version&status=any&order=asc&exact_name=phoneme&id=1489.

[7] Timo Freiesleben. 2022. The Intriguing Relation Between Counterfactual Explanations and Adversarial Examples. *Minds and Machines* 32, 1 (mar 2022), 77–109. https://doi.org/10.1007/s11023-021-09580-9

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research* 17, 1 (jan 2016), 2096–2030.

[9] Flavio Giorgi, Cesare Campagnano, Fabrizio Silvestri, and Gabriele Tolomei. 2025. Natural Language Counterfactual Explanations for Graphs Using Large Language Models. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 258)*, Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (Eds.). PMLR, 3565–3573. https://proceedings.mlr.press/v258/giorgi25a.html

[10] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local Rule-Based Explanations of Black Box Decision Systems. *arXiv preprint arXiv:1805.10820* (2018).

[11] Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nathan Srebro. 2018. Implicit Bias of Gradient Descent on Linear Convolutional Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NeurIPS '18)*. Curran Associates Inc., Red Hook, NY, USA, 9482–9491.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 630–645.

[13] Zecheng He, Tianwei Zhang, and Ruby B. Lee. 2019. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico, USA) *(ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 148–162. https://doi.org/10.1145/3359789.3359824

[14] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning Deep Representations by Mutual Information Estimation and Maximization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=Bklr3j0cKX

[15] Ziwei Ji and Matus Telgarsky. 2019. Gradient Descent Aligns the Layers of Deep Linear Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=HJflg30qKX

[16] Aditya Kadiwal. 2020. Water Potability Dataset. https://www.kaggle.com/datasets/adityakadiwal/water-potability Accessed: 2025-01-27.

[17] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 895–905.

[18] G.N. Karystinos and D.A. Pados. 2000. On Overfitting, Generalization, and Randomly Expanded Training Sets. *IEEE Transactions on Neural Networks* 11, 5 (2000), 1050–1057. https://doi.org/10.1109/72.870038

[19] V. Koltchinskii and D. Panchenko. 2002. Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers. *The Annals of Statistics* 30, 1 (2002), 1–50. http://www.jstor.org/stable/2700001

[20] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. (2009), 32–33. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[21] Thai Le, Suhang Wang, and Dongwon Lee. 2020. GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 238–248. https://doi.org/10.1145/3394486.3403066

[22] Romain Lopez, Adam Gayoso, and Nir Yosef. 2020. Enhancing Scientific Discoveries in Molecular Biology with Deep Generative Models. *Molecular Systems Biology* 16, 9 (2020), e9198. https://doi.org/10.15252/msb.20199198 arXiv:https://www.embopress.org/doi/pdf/10.15252/msb.20199198

[23] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. 2022. FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 5313–5322.

[24] Ana Lucic, Maartje A. Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. 2022. CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 151)*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.). PMLR, 4499–4511. https://proceedings.mlr.press/v151/lucic22a.html

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJzIBfZAb

[26] Maria Movin, Federico Siciliano, Rui Ferreira, Fabrizio Silvestri, and Gabriele Tolomei. 2025. Consistent Counterfactual Explanations via Anomaly Control and Data Coherence. *IEEE Transactions on Artificial Intelligence* 6, 4 (2025), 794–804. https://doi.org/10.1109/TAI.2024.3496616

[27] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 2022. Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis . In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 151)*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.). PMLR, 4574–4594. https://proceedings.mlr.press/v151/pawelczyk22a.html

[28] Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. 2011. Early Stopping for Non-Parametric Regression: An Optimal Data-Dependent Stopping Rule. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 1318–1325. https://doi.org/10.1109/Allerton.2011.6120320

[29] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. 2018. The Implicit Bias of Gradient Descent on Separable Data. *J. Mach. Learn. Res.* 19, 1 (Jan. 2018), 2822–2878.

[30] 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.

[31] David Stutz, Matthias Hein, and Bernt Schiele. 2019. Disentangling Adversarial Robustness and Generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6976–6987. https://doi.org/10.1109/CVPR.2019.00714

[32] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep Learning Face Representation from Predicting 10,000 Classes. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1891–1898. https://doi.org/10.1109/CVPR.2014.244

[33] Gabriele Tolomei and Fabrizio Silvestri. 2021. Generating Actionable Interpretations from Ensembles of Decision Trees. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2021), 1540–1553. https://doi.org/10.1109/TKDE.2019.2945326

[34] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. In *Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '17)*. Association for Computing Machinery, New York, NY, USA, 465–474. https://doi.org/10.1145/3097983.3098039

[35] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology*. 31 (2017), 841–887.

[36] David Warde-Farley, Ian J. Goodfellow, Aaron C. Courville, and Yoshua Bengio. 2014. An Empirical Analysis of Dropout in Piecewise Linear Networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6197

[37] Daniel Whiteson. 2014. HIGGS. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5V312.

[38] Kaiwen Wu and Yaoliang Yu. 2019. Understanding Adversarial Robustness: The Trade-off between Minimum and Average Margin. arXiv:1907.11780 [cs.LG] https://arxiv.org/abs/1907.11780

[39] Kevin Y. Yip and Mark Gerstein. 2008. Training Set Expansion: An Approach to Improving the Reconstruction of Biological Networks from Limited and Uneven Reliable Interactions. *Bioinformatics* 25, 2 (11 2008), 243–250. https://doi.org/10.1093/bioinformatics/btn602 arXiv:https://academic.oup.com/bioinformatics/article-pdf/25/2/243/48983145/bioinformatics_25_2_243.pdf

Table 4: Main properties and description of the datasets.

Dataset	#Samples	#Features	#Min. Class (%)	#Maj. Class (%)	Description
Water	3,276	9	1,278 (39%)	1,998 (61%)	This tabular dataset includes various water quality metrics and is used to predict water potability, i.e., whether it is safe for human consumption.
Phoneme	5,404	5	1,586 (29%)	3,818 (71%)	This tabular dataset contains five attributes selected to characterize each vowel, with the goal of distinguishing between nasal and oral sounds.
Higgs	200,000	28	94,131 (47%)	105,869 (53%)	This dataset is a random sample without replacement of the original Higgs tabular dataset, comprising 18% of the total 1.1 million instances, and is used to distinguish between a signal process that produces Higgs bosons and a background process that does not.
CIFAR-10	12,000	3×32×32	6,000 (50%)	6,000 (50%)	This dataset is a sample of the original CIFAR-10 image dataset, adapted to a binary classification setting, containing only instances from class 3 ("cats") and class 5 ("dogs").

A Additional Experimental Details

A.1 Datasets, Models, and Tasks

Three of the four datasets used in this study, namely Water, Phoneme, and Higgs, are tabular, while the fourth, CIFAR-10, is a popular image dataset. Table 4 summarizes key characteristics of each dataset, including the number of samples, number of features, class distribution (minority and majority class sizes), and a brief description.

All datasets except CIFAR-10 are naturally associated with a binary classification task. To adapt CIFAR-10 to a binary classification setting, we retain only instances from class 3 ("cats") and class 5 ("dogs").

For the Water, Phoneme, and Higgs datasets, we performed an 80/20 train-test split. For CIFAR-10, we used the standard train-test split provided by the torchvision library.

Table 5 summarizes the key characteristics of the models evaluated in our experiments. The selection covers a broad spectrum of architectures, from simple logistic regression models with a few thousand parameters to deep neural networks comprising tens of millions of parameters.

Table 5: Comparison of various models across datasets, layer configurations, and parameter counts.

Model	Dataset	Layers Width	#Parameters
LR	Water	N/A	5,005
$\text{MLP}_{\text{small}}$	Water	[100, 30]	3,930
$\text{MLP}_{\text{large}}$	Water	[150, 1000, 150, 30]	305,880
LR	Phoneme	N/A	4,368
$\text{MLP}_{\text{small}}$	Phoneme	[100, 40]	4,540
$\text{MLP}_{\text{large}}$	Phoneme	[150, 1000, 150, 30]	305,280
$\text{MLP}_{\text{small}}$	Higgs	[100, 30]	5,830
$\text{MLP}_{\text{large}}$	Higgs	[150, 1000, 150, 30]	308,730
CNN	CIFAR-10	ResNet-18	11,200,000

A.2 Training Settings

Table 6 illustrates the training settings used for each dataset. We adopt Adam as optimizer due to its faster convergence properties. This choice enables us to reduce the number of training epochs while still performing an exhaustive hyperparameter search within the same computational budget.

To illustrate Adam's faster convergence in practice, Figure 6 compares its performance to standard stochastic gradient descent (SGD) when training an MLP on the Higgs dataset.

Table 6: Training settings for each dataset.

Dataset	Optimizer	#Epochs	Batch Size	Learning Rate (η)
Water	Adam	2,000	128	0.001
Phoneme	Adam	2,000	128	0.001
Higgs	Adam	200	128	0.001
CIFAR-10	Adam	200	128	0.001

Training and testing are performed over 2,000 epochs for the Water and Phoneme datasets, and over 200 epochs for both CIFAR-10 and Higgs. In all cases, we use a batch size of 128, and we set the learning rate to $\eta = 0.001$ with no weight decay.

B Counterfactual Perturbation vs. Overfitting

In Section 4.4, we empirically demonstrated that the average ε-VCP is correlated with the risk of overfitting: specifically, higher training accuracy tends to correspond to a larger average ε-VCP. That analysis provided a general observation, with the average ε-VCP estimated using Monte Carlo integration.

In this section, instead, we conduct a more targeted analysis by examining the relationship between the test loss (i.e., the complement of test accuracy) and the average L_2-norm of the counterfactual perturbation vectors ($\overline{||\boldsymbol{\delta}||}$) over the course of training. The latter serves as a practical proxy for the average ε-VCP.

Table 7: Hyperparameter settings used to generate the results in Table 1. L1-Reg and L2-Reg report the regularization coefficients, i.e., the weights of the L_1 and L_2 norms of the model parameters, respectively. Dropout indicates the dropout probability. Early Stopping shows the patience parameter. PGD reports: the step size of the adversarial attack (α); the maximum perturbation budget (ϵ); and the number of attack iterations (k). Lastly, CF-Reg represents the regularization coefficients α and β.

Model	Dataset	L1-Reg	L2-Reg	Dropout	Early Stopping	PGD	CF-Reg (ours)
LR	Water	4.933 e-02	7.413 e-01	N/A	5	1.076 e-02/1.128 e-02/15	3.353 e-01/9.816 e-01
$\text{MLP}_{\text{small}}$	Water	1.280 e-02	2.920 e-03	2.004 e-01	15	1.364 e-01/4.714 e-02/05	8.325 e-01/1.886 e+00
$\text{MLP}_{\text{large}}$	Water	1.605 e-02	2.613 e-03	8.192 e-01	10	9.430 e-01/1.614 e-02/05	1.121 e+00/2.118 e+00
LR	Phoneme	1.796 e-02	3.659 e-02	N/A	260	1.543 e-02/6.009 e-03/05	1.303 e-05/3.461 e-02
$\text{MLP}_{\text{small}}$	Phoneme	1.097 e-03	6.972 e-03	1.606 e-01	280	1.100 e-02/9.069 e-02/20	1.485 e-02/1.883 e-01
$\text{MLP}_{\text{large}}$	Phoneme	5.885 e-03	3.003 e-03	1.001 e-02	260	1.044 e-02/7.132 e-03/05	5.539 e-03/1.797 e-01
$\text{MLP}_{\text{small}}$	Higgs	6.667 e-03	3.214 e-04	9.820 e-02	50	3.099 e-02/2.153 e-02/05	1.097 e-01/4.724 e-01
$\text{MLP}_{\text{large}}$	Higgs	1.391 e-02	1.675 e-04	3.918 e-01	5	1.422 e-02/8.403 e-02/05	4.380 e-01/2.289 e+00
CNN	CIFAR-10	8.700 e-06	4.868 e-06	N/A	160	3.755 e-01/5.043 e-03/07	7.246 e-04/1.852 e-01

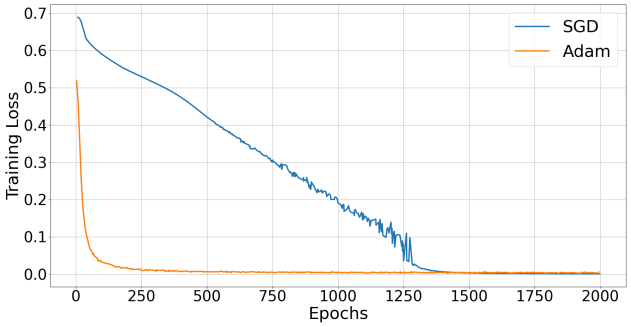


Figure 6: The training loss evolution over training epochs for an MLP trained on the Higgs dataset without regularization with SGD (blue) or Adam (orange) as optimizer. Adam shows a faster fit to training data in terms of training epochs.

Figure 7 illustrates the evolution of $\overline{||\boldsymbol{\delta}||}$ and the test loss over training epochs for an MLP trained without regularization on the Water dataset. The plot shows a clear trend as the model starts to overfit the data (evidenced by an increase in test loss). Notably, $\overline{||\boldsymbol{\delta}||}$ begins to decrease, which aligns with the hypothesis that the average distance to the optimal counterfactual example gets smaller as the model's decision boundary becomes increasingly adherent to the training data.

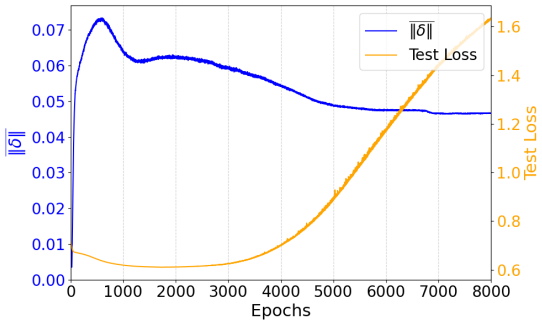


Figure 7: The average counterfactual perturbation vector $\overline{||\boldsymbol{\delta}||}$ (left y-axis) and the cross-entropy test loss (right y-axis) over training epochs (x-axis) for an MLP trained on the Water dataset *without* regularization.

It is worth noticing that this trend is heavily influenced by the choice of the counterfactual generator model. In particular, the relationship between $\overline{||\boldsymbol{\delta}||}$ and the degree of overfitting may become even more pronounced when leveraging more accurate counterfactual generators. However, these models often come at the cost of higher computational complexity, and their exploration is left to future work.

Nonetheless, we expect that $\overline{||\boldsymbol{\delta}||}$ will eventually stabilize at a plateau, as the average L_2-norm of the optimal counterfactual perturbations cannot vanish to zero.

C Hyperparameter Tuning

Most regularization techniques are highly sensitive to hyperparameter choices. Table 7 summarizes all hyperparameters used in this study. Each value was selected via random Bayesian optimization over approximately 80 trials, aiming to maximize accuracy on a held-out validation set.

In the table, L1-Reg and L2-Reg refer to the coefficients of the respective regularization terms, i.e., the weights of the L_1 and L_2 norms of the model parameters. The Dropout entry indicates the dropout probability applied within the model architecture. For Early Stopping, the reported value corresponds to the patience parameter, namely the number of epochs without improvement before training is stopped. The three entries listed under PGD correspond to: the step size of the adversarial attack (α); the maximum perturbation budget (ϵ); and the number of attack iterations (k). Finally, the two CF-Reg values represent the regularization coefficients α and β, as defined in (5) and (7), respectively.