Search Methods for Sufficient, Socially-Aligned Feature Importance Explanations with In-Distribution Counterfactuals

Peter Hase, Harry Xie, and **Mohit Bansal**
Department of Computer Science
University of North Carolina at Chapel Hill
{peter, fengyu.xie, mbansal}@cs.unc.edu

Abstract

Feature importance (FI) estimates are a popular form of explanation, and they are commonly created and evaluated by computing the change in model confidence caused by removing certain input features at test time. Two standard explanation metrics are *Sufficiency*, where only the top-k most important tokens are kept, and *Comprehensiveness*, where only the top-k important tokens are removed. In this paper, we study several under-explored dimensions of FI-based explanations, providing conceptual and empirical improvements for this form of explanation. First, we advance a new argument for why it can be problematic to remove features from an input when creating or evaluating explanations: the fact that these counterfactual inputs are out-of-distribution (OOD) to models implies that the resulting explanations are *socially misaligned*. The crux of the problem is that the model prior and random weight initialization influence the explanations (and explanation metrics) in unintended ways. To resolve this issue, we propose a simple alteration to the model training process, which results in more socially aligned explanations and metrics. Second, we compare among five approaches for removing features from model inputs. We find that some methods produce more OOD counterfactuals than others, and we make recommendations for selecting a feature-replacement function. Finally, we introduce four search-based methods for identifying FI explanations and compare them to strong baselines, including LIME, Integrated Gradients, and random search. On experiments with six diverse text classification datasets, we find that the only method that consistently outperforms random search is a Parallel Local Search that we introduce (as measured in terms of both standard FI metrics and our socially aligned metrics). Improvements over the second best method are as large as 5.4 points for Sufficiency and 17 points for Comprehensiveness.[1]

1 Introduction

Estimating feature importance (FI) is a common approach to explaining how learned models make predictions for individual data points [53, 47, 34, 60, 37, 15]. FI methods assign a single scalar to each feature of an input representing its "importance" to the model's output, where a feature may be an individual component of an input (such as a pixel or a word) or some combination of components. In parallel to the development of these methods, a variety of procedures have been proposed for evaluating FI estimates (also known as attributions) [43, 3, 17, 23, 22, 72]. Many of these procedures involve test-time ablations of the input, where features identified as important are removed from the

[1]All supporting code for experiments in this paper is publicly available at https://github.com/peterbhase/ExplanationSearch.

input, with the expectation that the model's confidence in its original prediction will decline as long as the selected features were truly important.

For instance, according to the *Sufficiency* metric [17], the best FI-based explanation is the set of features which, if kept, would result in the highest model confidence in its original prediction. Typically the top-k features would be selected according to their FI estimates, for some specified *sparsity* level k. Hence, the final explanation e is a k-sparse binary vector in $\{0,1\}^d$, where d is the dimensionality of the chosen feature space. As in [43, 17], Sufficiency should be computed at a specific explanation sparsity (or as an average across sparsity values) in order to control for how many features can be selected by an explanation. For an explanation e and a model f that outputs a distribution over classes $p(y|x) = f(x)$, Sufficiency can be given as:

$$\text{Suff}(f, x, e) = f(x)_{\hat{y}} - f(\texttt{Replace}(x, e))_{\hat{y}}$$

where $\hat{y} = \arg\max_y f(x)_y$ is the predicted class for x and the `Replace` function replaces features in x with some uninformative feature at locations corresponding to 0s in the explanation e.

The `Replace` function plays a key role in the definition of such metrics because it defines the *counterfactual* input that we are comparing the original input with. Though FI-based explanations are often presented without mention of counterfactuals, all explanations make use of counterfactual situations [39], and FI explanations are no exception. Indeed, the only way we can understand what makes some features "important" to a particular model prediction is by reference to a counterfactual input which has its important features replaced with a user-specified feature.

In this paper, we study several under-explored dimensions of the problem of finding good explanations according to test-time ablation metrics including Sufficiency and a related metric, Comprehensiveness, with a focus on natural language processing tasks. Our contributions cover (1) the problem of out-of-distribution counterfactuals in FI explanations, (2) which `Replace` functions are best to use, and (3) search-based methods for optimizing these metrics. We expand on these points below.

First, we argue that metrics like Sufficiency are heavily influenced by the out-of-distribution (OOD) nature of counterfactual model inputs. This leads to model explanations being heavily influenced the model prior and weight initialization, which results in *socially misaligned* explanations. We use this term, first introduced by Jacovi and Goldberg [25], to describe a situation where an explanation communicates a different kind of information than the kind that people expect it to communicate. Here, we do not expect our model prior or random weight initialization to influence FI estimates. This is a problem insofar as FI explanations are not telling us what we think they are telling us. We propose a training algorithm to resolve the social misalignment, which is to expose models to counterfactual inputs during training, so that counterfactuals are not out-of-distribution at test time.

Second, we systematically compare `Replace` functions, since this function plays an important role in evaluating explanations but no full comparison has been carried out. To do so, we remove tokens from inputs using several `Replace` functions, then measure how OOD these ablated inputs are to the model. We compare methods that remove tokens entirely from sequences of text [43, 17], replace token embeddings with the zero embedding or a special token [34, 60, 3, 70, 60], marginalize predictions over possible counterfactuals [73, 30, 68], and edit the input attention mask rather than the input text (our own proposal). Following our argument regarding the OOD problem (Sec. 4), we recommend the use of some `Replace` functions over others.

Third, we provide several novel search-based methods for identifying FI-based explanations. While finding the optimal solution to $\arg\max_e \text{Suff}(f, x, e)$ is a natural example of binary optimization, a problem for which search algorithms are a common solution [45, 50, 5], we are aware of only one prior work that searches for good explanations [21]. We introduce four novel search algorithms for finding good explanations by making use of general search principles [45]. Based on experiments with two Transformer models and six text classification datasets (including FEVER, SNLI, and others), we summarize our core findings as follows:

1. We propose to train models on explanation counterfactuals and find that this leads to greater model robustness against counterfactuals and yields drastic differences in explanation metrics.

2. We find that some `Replace` functions are much better than others, although ultimately our solution to the OOD problem is more effective at reducing counterfactual OOD-ness.

3. We introduce four novel search-based methods for identifying explanations. Out of all the methods we consider (including popular existing methods), the only one that consistently outperforms random search is the Parallel Local Search that we introduce (by margins of up to 20.8 points).

2 Related Work

Feature Importance Methods. A large number of methods have been introduced for FI estimation, drawing upon local approximation models [47, 48, 31], attention weights [26, 64, 71], model gradients [52, 51, 59, 54], and model-based feature selection [6, 4, 65, 44, 11, 15]. Recent work proposes to obtain explanations by selecting among explanations from several FI methods, since performance can vary across data points [12]. Surprisingly, we know of only one search technique that has been used to identify FI explanations: Fong and Vedaldi [21] perform gradient descent in explanation space. Some prior work even dismisses the approach on the grounds that the search problem is intractable [15]. However, search approaches are regularly used to solve combinatorial optimization problems in machine learning [50, 7, 5, 19, 42]. Here, we introduce several novel search algorithms for finding good explanations, including (1) a gradient search similar to Fong and Vedaldi [21], a (2) local heuristic search inspired by an adversarial attack method [19], (3) a global heuristic search, and (4) a parallel local search making use of general search principles [45].

Choice of `Replace` Function. To our knowledge, no prior work systematically compares the effect of the `Replace` function on metric results for a fixed set of explanations. Past evaluations of explanation methods typically remove tokens or words from the text entirely [43, 17] or replace them with fixed feature values [23, 69]. Methods for creating explanations also use several distinct approaches, including (1) replacing token embeddings with the zero vector [34, 60, 3], (2) using a special token [70, 60], (3) marginalizing predictions over a random variable representing an unobserved token [73, 30, 68, 28], and (4) adversarially selecting counterfactual features [24]. Sturmfels et al. [57] carry out a case study involving a few `Replace` functions for a vision model, but they focus specifically on image blurring techniques from Fong and Vedaldi [21] and the case study is not intended to be a full comparison of methods. In addition to evaluating `Replace` functions from the above works, we also consider setting attention mask values for individual tokens to 0, which reduces their attention weights to 0 at each layer in a Transformer.

The Out-of-distribution Problem of Explanations. Many papers have expressed concerns over how removing features from an input may result in the input being out of distribution to a trained model [70, 60, 21, 10, 23, 30, 24, 68, 58, 28]. In response to the problem, some have proposed to marginalize model predictions over possible counterfactual inputs [73, 30, 68, 28] or to adversarially select counterfactual features rather than use any user-specified features [24]. Others reject the whole notion of test-time ablations, preferring metrics based on train-time ablations [23]. Prior works typically make arguments for their approach based on intuition or basic machine learning principles, such as avoiding distribution shift. In Sec. 4, we give a more detailed argument for preferring in-distribution counterfactuals on the basis of *social alignment*, a concept introduced by Jacovi and Goldberg [25], and we propose a solution to the OOD problem. Our solution allows for test-time evaluation of explanations for individual data points, unlike the most similar proposal, which is based on train-time ablations and can only evaluate large sets of explanations all at once [23].

3 Problem Statement

Feature Importance Metrics. The problem we are investigating is to find good feature importance explanations for single data points, where explanation are evaluated under metrics using test-time ablations of the input. In this context, an explanation for an input in a d dimensional feature space is a binary vector $e \in \{0,1\}^d$, which may be derived from discretizing an FI estimate $v \in \mathbb{R}^d$. We consider two primary metrics, *Sufficiency* and *Comprehensiveness* [17]. Sufficiency measures whether explanations identify a subset of features which, when kept, lead the model to remain confident in its original prediction for a data point. Comprehensiveness, meanwhile, measures whether an explanation identifies all of the features that contribute to a model's confidence in its prediction, such that removing these features from the input lowers the model's confidence. Both of these metrics technically compute the same quantity: the difference between the model output for an input and the output for an ablated version of the input. But the sparsity of explanations differs between them, and it is intended for one to be maximized while the other is minimized.

The Sufficiency metric for an explanation $e \in \{0,1\}^d$ and model $p(y|x) = f_\theta(x)$ is given as

$$\text{Suff}(f_\theta, x, e, s) = f_\theta(x)_{\hat{y}} - f_\theta(\texttt{Replace}_s(x, e))_{\hat{y}} \quad (1)$$

where $\hat{y} = \arg\max_y f(x)_y$ is the predicted class for x, and the $\texttt{Replace}_s$ function retains a proportion s of the input features (indicated by e) while replacing the remaining features with some user-specified feature. In order to control for the explanation sparsity, i.e. the proportion s of tokens in the input that may be retained, we average Sufficiency scores across sparsity levels in {.05, .10, .20, .50}, meaning between 5% and 50% of tokens in the input are retained. These values follow from DeYoung et al. [17], with the .01 value omitted since it is equivalent to using .05 for inputs that are at most 20 tokens long. A **lower** Sufficiency value is better, as it indicates that more of the model's confidence is explained by just the retained features (increasing $f_\theta(\texttt{Replace}(x, e))_{\hat{y}}$).

Similarly, Comprehensiveness is given as

$$\text{Comp}(f_\theta, x, e, s) = f_\theta(x)_{\hat{y}} - f_\theta(\texttt{Replace}_s(x, e))_{\hat{y}} \tag{2}$$

but with sparsity values in {.95, .90, .80, .50}, as we are looking to remove features that are important to the prediction (while keeping most features). A **higher** Comprehensiveness value is better, as it indicates that the explanation selects more of the evidence that contributes to the model's confidence in its prediction (resulting in a lower $f_\theta(\texttt{Replace}(x, e))_{\hat{y}}$).

Overall Objective. Finally, our overall Sufficiency and Comprehensiveness objectives are given by averaging Suff (or Comp) scores across several sparsity levels. With a model $p(y|x) = f(x)$, a single data point x with d features, and a set of sparsity levels S, the Sufficiency objective is optimized by obtaining a set $E = \{e_i\}_{i=1}^{|S|}$ with one explanation per sparsity level as

$$\begin{aligned} &\arg\max_E \frac{1}{|S|} \sum_{i=1}^{|S|} \text{Suff}(f, x, e_i, s_i) \\ &\text{s.t. } e_i \in \{0, 1\}^d \text{ and} \\ &\sum_d e_i^{(d)} \leq \text{ceiling}(s_i \cdot d) \end{aligned}$$

where the ceiling function rounds up the number $s_i \cdot d$ of tokens to keep. When optimizing for Comprehensiveness, we use the Comp and $\arg\min$ functions, and the inequality is flipped.

Controlling for Compute Budget. We wish to control for the compute budget used to obtain E in order to fairly compare explanation methods. Some explanations require a single forward and backward pass using f_θ [53, 33], while others can require hundreds of forward and backward passes [60] or thousands of forward passes [47]. Since computing f_θ can be expensive, we limit each method in our study to a fixed number of forward and backward passes (counted together), typically 1000 in total (further described in Sec. 7).

Weight of Evidence Formulation. Note that we can also measure either metric in terms of a difference in log-odds (i.e. weight of evidence) rather than probabilities, which reflects differences in evidence for classes before this evidence is compressed to the bounded probability scale [49, 2]. In this case, Sufficiency e.g. is computed as

$$\text{Suff}_{\text{WoE}}(f_\theta, x, e) = \ln \frac{f_\theta(x)_{\hat{y}}}{f_\theta(x)_{\neg\hat{y}}} - \ln \frac{f_\theta(\texttt{Replace}(x, e))_{\hat{y}}}{f_\theta(\texttt{Replace}(x, e))_{\neg\hat{y}}} \tag{3}$$

where $f_\theta(x)_{\neg\hat{y}}$ is the sum of all class probabilities except the predicted class, meaning each term is the log-odds in favor of $\hat{y}$. In Sec. 7 we describe results for the standard difference-in-probabilities version of each metric as well as the weight-of-evidence versions.

4 The Out-of-Distribution Problem in Explanations

In this section, we first give a full argument for why it is problematic for explanations to be created *or* evaluated using OOD counterfactuals. Next, we propose a simple solution to the OOD problem. We rely on this argument in our comparison of `Replace` functions in Sec. 5, which motivates salience-based and search-based explanation method design choices in Sec. 6. We also assess our proposed solution to the OOD problem in Sec. 5 and later make use of this solution in 7.

The OOD problem for explanations occurs when a counterfactual data point used to create or evaluate an explanation is out-of-distribution (OOD) to the model. Here, we take OOD to mean that the data point is not drawn from the data distribution used in training [41]. We focus on the OOD problem for creating explanations when giving our argument, but the argument covers explanation evaluation as well. In general, counterfactual data points used to produce FI explanations will be OOD because they contain feature values not seen in training, like a MASK token for a language model. A long line of past work raises concerns with this fact: Zaidan et al. [70] note that using OOD counterfactual features can cause "feature extraction to behave strangely." Sundararajan et al. [60] and Fong and Vedaldi [21] suggest that using OOD inputs could lead to attribution "artifacts," while Yi et al. [68] comment that OOD inputs result in "ill-defined [model] behavior." Kim et al. [30] give a more specific argument, asserting that OOD counterfactuals cause the feature importance of an observed feature to be "overestimated" because models tend to assign lower predicted probabilities to OOD inputs. Meanwhile, Hsieh et al. [24] argue that both user-specified counterfactual features and marginalization approaches introduce an "intrinsic bias" to explanation evaluation, resulting either from the choice of counterfactual feature or the biases of the generative model used for marginalization. The closest argument to our own is presented by Hooker et al. [23], who observe that using OOD counterfactuals makes it difficult to determine whether model performance degradation is caused by "distribution shift" or by the removal of information. Similar to the goal of ROAR [23], our proposed solution to the OOD problem also aims to align the train and test-time distributions, though our approach still allows for test-time evaluation of individual explanations. We next clarify what exactly makes distribution shift undesirable in the context of feature importance explanations.

Our principal claim is that feature importance explanations for a standardly trained neural model are *socially misaligned*, which is undesirable. Jacovi and Goldberg [25] originally introduce this term as they describe shortcomings of explanations from a pipeline model, which is a kind of classifier designed to be interpretable. They observe that people expect these models to make decisions in one manner, while the model actually makes decisions in some other manner and explanations fail to alert them to this fact. Here, we use *social misalignment* to describe how people's expectations of what FI explanations will communicate are not fulfilled by what they actually communicate. Our argument now proceeds in two steps: first, we outline what the social expectations are for feature importance explanations, and then we argue that the social expectations are violated due to the OOD problem.

We suggest that, for a particular trained model and a particular data point, **people expect a feature importance explanation to reflect how the model *has learned* to interpret features as evidence for or against a particular decision.**[2] This social expectation is upheld if FI explanations are influenced only by the interaction of an untrained model with a training algorithm and data. But our expectations are violated to the extent that FI explanations are influenced by factors such as the choice of model prior and random seed (which we do not intend to influence FI explanations). We depict these possible upstream causes of individual FI explanations in Fig. 1.

In fact, the model prior and random seed are influential to FI explanations when the counterfactuals employed in these explanations are OOD to the model. A simple example clearly illustrates the potential influence of model priors: predictions from a Gaussian Process model in regions of the data domain lacking training data will be highly dependent on the user's choice of mean and covariance functions (see Fig. 4 in Appendix C for visual examples) [8, 62]. If one were to estimate feature importance for a GP model using OOD data as a counterfactual input, the resulting FI estimates would be extremely sensitive to the user's priors.

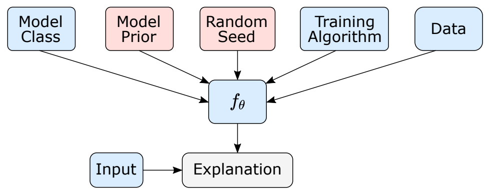

Figure 1: Causal diagram of a feature importance explanation for a particular trained model and input. Explanations are socially misaligned when the model prior and random seed influence explanations in unexpected ways.

In the case of neural language models, we know that models are sensitive to random parameter initialization and data ordering (determined by the random seed) [18] and hyperparameters (including regularization coefficients) [13, 46, 63], even as evaluated on in-distribution data. When it comes to OOD data, the model will be still influenced by these factors, but the model has no data to learn from in this domain. As a result, FI explanations are socially misaligned to the extent that these unexpected factors influence the explanations while the expected factors, like data, are less influential.

The solution to the OOD problem is to **align the train and test distributions, which we do by exposing the model to counterfactual inputs during training**, a method we term Counterfactual

[2]We mean "people" to refer to the typical person who has heard the standard description of these explanations, i.e. that they identify features that are "important" to a model decision.

Training. Yet, common explanation methods can require hundreds or thousands of model forward passes when explaining predictions [47, 60]. Since this means that explanations would be prohibitively expensive to obtain during training, we propose to train with random explanations that remove most of the input tokens. Random explanations can be cheaply produced while assigning uniform probability to counterfactuals that will be seen at test time. Moreover, applying the `Replace` function with random explanations will typically yield more difficult training inputs than, for example, what explanations for Sufficiency might yield, as removing a large proportion of random tokens will tend to leave only a small amount of label information in the input. We leave data labels unaltered, since we have no reason a priori to alter them for ablated inputs. In practice, we double the inputs in each training batch, adding a `Replace`(x, e) version of each input (with the same label) according to a random explanation e with sparsity randomly selected from $\{.05, .1, .2, .5\}$. The resulting Counterfactual-Trained (CT) models make in-distribution predictions for both observed and counterfactual data points, resolving the OOD problem and social misalignment (though random seeds may still influence FI estimates in an unexpected manner). We find that these models suffer only slight drops in test set accuracy, by 0.7 points on average across six datasets (see Table 5 in Appendix C). But we observe that this approach greatly improves model robustness to counterfactual inputs, meaning the counterfactuals are much more in-distribution to models (described in Sec. 5). In Sec. 7, we also find sizable differences in explanation metrics depending on whether we use Counterfactual-Trained models, which implies that standard FI estimates and FI metrics can be heavily influenced by unexpected factors like the model prior and random seed.

We note that the Remove-and-Retrain procedure of Hooker et al. [23] also aligns the train and test distributions in order to resolve the OOD problem, although our proposal alters the one-time training process, while ROAR requires retraining one model per set of explanations (e.g. one per explanation sparsity level). Additionally, we are still primarily interested in explaining the behavior of already-trained models though test-time ablations for single data points, whereas ROAR evaluates a large set of explanations all at once and measures feature importance for a model class rather than particular trained models.

5 Analysis of Counterfactual Input OOD-ness

In this section, we assess how out-of-distribution the counterfactual inputs given by `Replace` functions are to models, and we measure the effectiveness of our Counterfactual Training scheme. We do this before designing or evaluating explanation methods because, given our argument in Sec. 4, it is important to first identify which `Replace` function and training methods are most appropriate to use for these purposes.

Experiment Design. We compare between `Replace` functions according to how robust models are to test-time input ablations using each function, where the set of input features to be removed is fixed across the functions. We measure robustness by model accuracy, which serves as a proxy for how in-distribution or out-of-distribution the ablated inputs are. If we observe differences in model accuracies between `Replace` functions for a given level of feature sparsity, we can attribute the change in the input OOD-ness to the `Replace` function itself. In the same manner, we compare between Counterfactual-Trained models and standardly trained models (termed as Standard).

Specifically, we train 10 BERT-Base [16] and RoBERTa-Base [35] on two benchmark text classification datasets, SNLI [9] and SST-2 [56]. These are all Standard models, without the counterfactual training we propose. We use ten random seeds for training these models. Then, we evaluate how robust the models are to input ablations, where we remove a proportion of random tokens using one of five `Replace` functions (i.e. we `Replace` according to a random explanation). We evaluate across proportions in $\{0.2, 0.5, 0.8\}$. The five `Replace` functions we test are:

1. **Attention Mask.** We introduce this `Replace` function, which sets a Transformer's attention mask values for removed tokens to 0, meaning their hidden representations are never attended to. To make this a differentiable function, we compute the function by taking the element-wise product between the attention distribution and the binary attention mask, then renormalizing the attention probabilities to sum to 1. The difference between this approach and deleting a token from an input text is that positional embeddings for retained tokens are unchanged.
2. **Marginalize.** This method obtains an expected model prediction over counterfactuals drawn from a generative model of the data $p_\phi(x)$, e.g. a BERT model [73, 30, 10, 68]. We replace

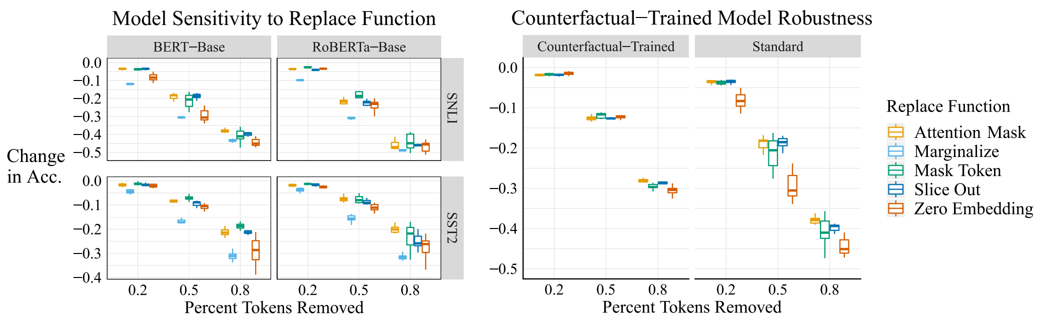

Figure 2: Model sensitivity to input ablations for several choices of `Replace` function and training algorithm. On the left we show the sensitivity of standardly trained models. On the right we show the effect of using Counterfactual-Trained models.

tokens in x with samples from the generative model, with the goal of producing in-distribution counterfactuals. As in Kim et al. [30], we use a pretrained BERT-Base as our generative model (or a RoBERTa-Base model, when the classifier is a RoBERTa model). To impute all of the tokens to be removed, we first replace each token with a MASK token, then impute one at a time in a random order by randomly sampling a replacement token. The final prediction is obtained from the marginal log probability as $\arg\max_y \ln \sum_{\tilde{x}\sim p_\phi(\tilde{x}|x,e)} p_\theta(y|\tilde{x})p_\phi(\tilde{x}|x,e)$, where $p_\phi(\tilde{x}|x,e)$ is the distribution over imputed tokens. Since computing this marginal distribution is quite expensive, we adopt a Monte Carlo approximation common to past work [30, 68]. Similar to Yi et al. [68], we find that approximation quality plateaus after a small number of samples (see Appendix A for tuning details), which allows us to evaluate this approach at scale.

3. **MASK Token**. In this approach, we simply replace tokens with the MASK token.
4. **Slice Out**. This approach removes selected tokens from the input sequence itself, such that the resulting sequence has a lower number of tokens. We delete tokens after tokenizing the input string (i.e. mapping from the input string to a sequence of tokens).
5. **Zero Vector**. Here, we set the token embeddings for removed tokens to the zero vector.

We train additional CT models for BERT-Base on SNLI, with ten random seeds per model, for all `Replace` functions except Marginalize, since this function is exceedingly expensive to use during Counterfactual Training due to the required Monte Carlo sampling and model-based imputation.

In this evaluation, the tokens to be hidden from the model are selected uniformly at random without replacement. We sample 10 random masks per data point and take the majority prediction on the corresponding 10 ablated data points as the model's overall prediction. The exception to this procedure is Marginalize, since it is much more computationally expensive than other methods, and therefore we use only one random explanation per input. Note that we use a subset of 10k train points for each task, and we perform this experiment on validation splits because the experiment motivates method design choices in Sec. 6.

Results for `Replace` functions. We show the results of this experiment in Fig. 2, via boxplots of the drops in accuracy for each of the 10 models per condition. First, we describe differences in `Replace` functions for Standard models, then we discuss the effect of Counterfactual Training.

On the left hand side of Fig. 2, we see that models are much more sensitive to some `Replace` functions than others. We report a few main conclusions, conducting hypothesis tests[3] on differences between methods with 50% or 80% of tokens removed: (1) The Attention Mask and Mask Token functions are the two best methods across models and datasets. The best of these two methods outperforms the third best method by up to 1.61 points with BERT and SNLI ($p = .0005$), 5.48 points with RoBERTa and SNLI ($p <$ 1e−4), 2.42 points with BERT and SST-2 ($p = 0.0008$), and 4.72 points with RoBERTa and SST-2 ($p <$ 1e−4). (2) While the Marginalize method is intended to produce in-distribution counterfactuals, it actually produces the most OOD counterfactuals across model classes and datasets, with a few drops in accuracy greater than 10 points compared to the best `Replace` function. (3) The Zero Embedding and Slice Out functions often far underperform

[3] p-values for two-sided difference in means tests are calculated by block bootstrap with all data points and model seeds being resampled 100k times [20].

the best Replace function. For instance, with BERT on SST-2, Zero Embedding is up to 10.45 points worse than Mask Token ($p < 1\text{e}{-4}$), and with RoBERTa on SST-2, Slice Out underperforms Attention Mask by up to 4.72 points ($p < 1\text{e}{-4}$). Overall, **we recommend that, when not using Counterfactual Training, researchers use either the Attention Mask or Mask Token Replace functions**. We recommend comparing these two methods for the particular trained model being explained because, due to seed variance, one method may outperform the other depending on the random seed used in training. For example, when using RoBERTa with SST-2 and removing 80% of tokens, Attention Mask outperforms Mask Token by 3.06 points ($p = .0459$), but Attention Mask scores are actually higher for only 6 of the 10 training seeds, with the other 4 favoring Mask Token.

Counterfactual Training vs. Standard Training. On the right hand side of Fig. 2, we see the effect of Counterfactual Training on model robustness for several Replace functions. We find that counterfactual inputs are much less OOD for Counterfactual-Trained models than for Standard models, regardless of the Replace function used. The improvement in robustness is up to **22.9** points. Moreover, the difference between Replace functions is almost entirely erased, though we do observe a statistically significant difference between Attention Mask and Zero Embedding with 80% of tokens removed (by 2.23 points, $p < 1\text{e}{-4}$). Given these results, **we ultimately recommend that researchers use Counterfactual Training with the Attention Mask, Mask Token, or Slice Out Replace function whenever they intend to create FI-based explanations.**

6 Methods

In this section, we describe the existing explanation methods we use as well as the methods we introduce, divided up into two broad categories: salience methods and search methods.

6.1 Saliency Methods

One family of approaches we consider assigns a scalar *salience* to each feature of an input. While salience scores may represent a variety of feature properties (such as an "effect on the output" or "probability of importance"), the key property of these scores is that they allow one to rank-order features. We obtain binarized explanations through selecting either the top-k features or up to the top-k features when some scores are negative and the model output is being maximized (or vice versa). Appendix A contains additional implementation details for each method below. Compute budgets are described later in Sec. 7.1.

1. LIME. LIME estimates FI by learning a linear model of a model's predicted probabilities with samples drawn from a local region around an input [47]. Though it is common to use the Slice Out Replace function with LIME, we use the Attention Mask Replace function (following Sec. 5), meaning we sample local attention masks rather than local input sequences. For a data point x, we train a model m_ϕ minimizing an MSE weighted by the kernel π and regularized by Ω,

$$\sum_{i=1}^{N} \pi(x, \tilde{x}_i)(m_\phi(\tilde{x}_i) - f_\theta(\tilde{x}_i)_{\hat{y}})^2 + \Omega(\phi)$$

where $f_\theta(x)_{\hat{y}}$ is the task model's predicted probability, local samples $\tilde{x}_i$ have attention masks that are imputed with a random number of 0s, and Ω is the default auto regularization in the LIME package.

2. Vanilla Gradients. We evaluate the use of model gradients w.r.t. the model input as salience scores, since this method has been relatively successful under other kinds of evaluations [1]. Here, we compute the gradient of the model's predicted probability w.r.t. the input token embeddings (of shape $L \times D$ for a document with L tokens and D-dimensional embeddings), and we obtain a single value per token by summing along the embedding dimension. This is the only method which does not have a compute budget, as it requires only a single forward and backward pass.

3. Integrated Gradients. We evaluate the Integrated Gradients (IG) method of Sundararajan et al. [60]. This method numerically integrates the gradients of a model output w.r.t. its input along a path between the observed input and a user-selected baseline. The salience for an input x with baseline $\tilde{x}$ is given as

$$(x - \tilde{x}) \times \int_{\alpha=0}^{1} \frac{\partial f(\tilde{x} + \alpha(x - \tilde{x}))}{\partial x} d\alpha.$$

We use the input embeddings of a sequence as x. Given our results in Sec. 5, we use a repeated MASK token embedding for our baseline $\tilde{x}$ rather than the all-zero input suggested by Sundararajan et al. [60] for text models. We use the model's predicted probability as the output, and to obtain token-level salience scores, we sum the output of IG along the embedding dimension. By the Completeness property of IG, token-level salience scores still sum to the difference in predicted probabilities between the observed input and the baseline.

6.2 Search Methods

An alternative class of methods searches through the space of possible explanations. Search methods are regularly used to solve combinatorial optimization problems in machine learning [50, 7, 5, 19, 42], and they may be applied here to optimize our objective. We list the search methods and some core properties in Table 1. Note that all search methods use the Attention Mask `Replace` function, and the search space is always restricted to explanations of the maximum allowable sparsity (or minimum, with Comprehensiveness).

	Search Method Properties	
Method	Computes ∇f	Proposal Fn.
Random	✗	Global
Exhaustive	✗	Global
Gradient	✓	Local
Taylor	✓	Local
Ordered	✗	Global
Parallel Local	✗	Local

Table 1: Search Method Properties. Computes ∇f indicates that the method requires computing model gradients. Proposal Fn. refers to how the current state is updated to a new proposed state.

1. Random Search. We use random search as a baseline for other methods. For each maximum explanation sparsity k (or minimum, for Comprehensiveness), we randomly sample a set of k-sparse explanations, compute the current objective for each of them, and choose the best explanation under the objective.

2. Exhaustive Search. Exhaustive search returns the optimal solution after checking the entire solution space. This is prohibitively expensive with large inputs, as there is a combinatorial explosion in the number of possible explanations. However, we are able to exactly identify optimal explanations for short sequences, typically with 10 or fewer tokens, so we use this method in limited settings.

3. Gradient Search. Whereas Fong and Vedaldi [21] propose to search through a continuous explanation space by gradient descent, we introduce a Gradient Search that uses discrete explanations. We enforce discreteness because continuous explanations produce OOD inputs, since they let some fractional amount of the input through to the model. They also do not reflect test-time conditions where discrete explanations must be used. For an input of length L, this method sequentially updates a continuous state vector $s \in \mathbb{R}^L$ via gradient descent in order to minimize a regularized cross-entropy loss between the original model prediction $\hat{y}$ and the predicted probability given the input $\tilde{x} = \texttt{Replace}(x, e_t)$. The explanation e_t is sampled as follows: $e^{(d)} \sim \text{Gumbel-Softmax}(s^{(d)})$, for $d = 1 : L$. The new state is $s_{t+1} \leftarrow s_t - \alpha \nabla_s \mathcal{L}(\hat{y}, f(\tilde{x})_{\hat{y}})$, though note that we use an AdamW optimizer for this step. By virtue of the differentiable Attention Mask `Replace` function and the Gumbel-Softmax estimator [38, 27], this loss is differentiable w.r.t. s. The regularizer is an ℓ_2 penalty on the difference between the expected sparsity $\sum_{d=1}^{L} \sigma(s^{(d)})$ and a target sparsity, set to ceiling$(.05 \cdot L)$ in order to encourage searching through sparse explanations. The final explanation is obtained by taking the top-k values of the last state s_t according to the maximum sparsity k.

We observe that this search method is equivalent to fitting a non-parametric model to the dataset with the objective $\mathcal{L}$ given above. Recently, many parametric models have been proposed for sampling explanations for individual data points [6, 4, 65, 44, 11, 15]. In early experiments, we found that a parametric model performed far worse than this non-parametric approach, and hence we leave out parametric models from further consideration. This is perhaps unsurprising given how hard it may be to learn a map from inputs to good explanations for all data.

4. Taylor Search. Inspired by HotFlip [19], this method explores the solution space by forecasting the change in the objective using a first-order Taylor approximation. At time step t, the state is an explanation e_t, and a heuristic is evaluated on neighboring states in order to select the next state to compute the objective on. The search space is all k-sparse explanations, and therefore neighboring states are those with Hamming distance 2 to the current state (with one retained token being hidden and one hidden token being retained). The heuristic is the projected change in the cross-entropy loss between the model's original prediction $\hat{y}$ and this label's probability given the input $\texttt{Replace}(x, e)$, for a proposed explanation e, which is computed as such: We first calculate the gradient $g \in \mathbb{R}^L$ of the cross-entropy loss with respect to the explanation e_t, which is possible with the differentiable Attention Mask `Replace` function. Then, when optimizing for Sufficiency, we find the two indices i and j as the solution to $\arg\max_{i,j}\ g^{(i)} - g^{(j)}$ such that the sparsity is maintained by flipping both tokens, meaning $e^{(i)} = 1$ ($x^{(i)}$ is retained) and $e^{(j)} = 0$ ($x^{(j)}$ is hidden). The next state is obtained by setting $e_t^{(i)}$ and $e_t^{(j)}$ to these values. This is a first-order approximation to the change in loss between the new and old state, based on the Taylor expansion of the loss [19]. Note when optimizing for Comprehensiveness, we use the $\arg\min$. Following Ebrahimi et al. [19], we ultimately use this search heuristic within a beam search, starting from a random explanation, with width $w = 3$.

5. Ordered Search. Next, we introduce a global search algorithm, Ordered Search. The premise of Ordered Search is to search the entire solution space in order of how promising individual solutions are, as estimated by an initial salience-based FI estimate. That is, we search through all explanations in an order given by a scoring function $f : e \rightarrow \mathbb{R}$. We only require that f is linear in e, as this allows for efficient ordering of the search space using a priority queue. More complicated forms for f, such as being quadratic in e, would make finding the optimum of the function computationally intractable, let alone a full rank-ordering of solutions [5]. The algorithm proceeds in two stages: (1) estimating parameters for f_θ, and (2) searching through explanations in order of their score, $\theta^T e$. For the first stage, we obtain parameters for f_θ from the per-token salience scores given by LIME, which is the best salience method evaluated in Sec. 7. In the second stage, we enumerate the search space (up to the remaining compute budget) in order of the score given by f_θ. Compute budget (measured in terms of forward passes) is allocated between these two stages according to tuning results (see Appendix A); the first stage consumes 25% of the budget, and the second stage 75%.

6. Parallel Local Search. Lastly, we consider the class of gradient-free local search algorithms, which have a long history of success with combinatorial optimization problems [45, 5]. We propose a parallelized local search algorithm tailored to the problem at hand. Given a number r of parallel runs to perform, a proposal function `Propose`, and compute budget b per run, an individual search proceeds as follows:

1. Sample an initial explanation e_1 randomly from the solution space and compute the objective function for that explanation.
2. For the remaining budget of $b-1$ steps: sample a not-already-seen explanation e_t according to `Propose` and adopt e_t as the new state only if the objective is lower at e_t than at the current state.

The `Propose` function samples new explanations by starting a random walk from the current explanation that ends when a not-already-seen explanation is encountered. As in Taylor Search, neighboring states have Hamming distance 2. Though we parallelize this search method, we maintain a shared set of previously-seen explanations and compute the `Propose` function serially at each step so that we never compute the more expensive objective function on the same explanations. This algorithm performs favorably to alternatives that have a probabilistic update rule (like Simulated Annealing) as well as non-parallelized versions with larger budgets b (see Appendix A). Following tuning results, we use $r = 10$ parallel runs for all experiments, meaning that each run has a budget $b = 100$ when the overall method budget is 1000 forward passes.

7 Explanation Method Experiments

7.1 Setup

Data. We compare the above explanation methods on six benchmark text classification datasets. Five are shared with the ERASER dataset suite [17]: SNLI [9], BoolQ [14], Evidence Inference [32], FEVER [61], and MultiRC [29]. Additionally, we include SST-2 [55] due to the availability of short sentences in this dataset, which allows us to compare with exhaustive search. One important distinction among these datasets is that the BoolQ, FEVER, MultiRC, and Evidence Inference data points include both a *query* and an accompanying *document*. The query is typically critical information that indicates how the model should process the document, and therefore we never `Replace` query tokens at any time. Note that we also never replace special tokens, such as the CLS token in BERT whose final representation is used for classification. In our experiments, we use 500 test points from each dataset to compare methods, selecting the shortest sequences in the test set for SST-2. See Table 4 in Appendix C for dataset statistics, including average sequence length.

		Sufficiency ↓		Comprehensiveness ↑	
Dataset	Method	Standard Model	CT Model	Standard Model	CT Model
SNLI	LIME	20.00 (2.02)	27.08 (1.68)	82.18 (2.82)	75.34 (1.93)
	Int-Grad	43.76 (3.27)	32.91 (2.36)	34.01 (2.55)	43.22 (2.28)
	Gradient Search	17.55 (1.47)	33.98 (1.43)	53.15 (2.53)	49.36 (1.95)
	Taylor Search	6.91 (1.10)	28.00 (1.46)	73.20 (2.57)	66.76 (2.12)
	Ordered Search	-1.45 (0.93)	15.06 (1.37)	87.78 (2.41)	84.67 (1.61)
	Random Search	-1.54 (0.96)	15.38 (1.39)	87.36 (2.47)	84.63 (1.68)
	Parallel Local Search	**-1.65** (1.07)	**14.16** (1.38)	**87.95** (2.55)	**86.18** (1.45)
BoolQ	LIME	2.15 (1.75)	-1.56 (0.63)	52.02 (3.69)	36.25 (3.45)
	Int-Grad	20.78 (3.57)	9.05 (1.53)	16.80 (1.57)	12.20 (1.68)
	Gradient Search	5.12 (1.41)	1.65 (0.81)	30.04 (2.58)	17.65 (1.85)
	Taylor Search	6.01 (1.33)	2.28 (0.87)	46.32 (3.89)	26.65 (2.68)
	Ordered Search	0.09 (0.84)	-2.58 (0.70)	51.59 (3.52)	34.36 (3.34)
	Random Search	-0.58 (0.63)	-2.51 (0.70)	55.78 (3.71)	31.62 (3.06)
	Parallel Local Search	**-1.17** (0.47)	**-3.52** (0.88)	**72.78** (4.06)	**47.80** (3.57)
Evidence Inference	LIME	-16.07 (2.84)	-14.92 (1.38)	47.60 (5.66)	33.97 (4.22)
	Int-Grad	1.22 (4.42)	-2.98 (1.68)	26.51 (2.68)	20.87 (2.57)
	Gradient Search	-10.57 (2.58)	-7.56 (1.46)	31.73 (4.43)	18.07 (2.13)
	Taylor Search	-4.55 (2.66)	-3.33 (1.27)	41.95 (5.63)	26.70 (3.00)
	Ordered Search	-16.80 (2.75)	-14.26 (1.36)	45.37 (5.53)	31.14 (3.73)
	Random Search	-17.05 (2.83)	-12.69 (1.30)	42.81 (6.00)	26.48 (3.15)
	Parallel Local Search	**-20.76** (3.77)	**-20.33** (2.65)	**56.31** (9.81)	38.71 (3.91)
FEVER	LIME	-0.24 (0.50)	0.39 (0.96)	33.86 (3.43)	22.06 (2.36)
	Int-Grad	9.72 (1.80)	4.99 (1.40)	17.81 (2.47)	13.69 (1.71)
	Gradient Search	0.66 (0.68)	2.63 (1.12)	19.26 (2.68)	11.44 (1.65)
	Taylor Search	4.17 (0.96)	4.20 (1.20)	24.51 (2.78)	15.62 (1.85)
	Ordered Search	-1.26 (0.41)	-0.01 (0.90)	31.79 (3.28)	18.90 (2.46)
	Random Search	-1.51 (0.51)	-1.24 (2.33)	32.47 (3.33)	18.84 (2.11)
	Parallel Local Search	**-2.04** (0.62)	**-3.66** (0.82)	**37.72** (3.28)	**24.07** (2.46)
MultiRC	LIME	-5.20 (1.18)	-5.90 (1.19)	39.75 (4.84)	**28.57** (2.18)
	Int-Grad	13.19 (3.14)	4.66 (1.71)	15.53 (3.39)	11.84 (1.31)
	Gradient Search	-0.09 (1.33)	-0.73 (1.18)	20.16 (2.92)	11.41 (1.13)
	Taylor Search	7.54 (2.53)	1.43 (1.47)	30.76 (4.04)	20.15 (1.83)
	Ordered Search	-6.43 (0.98)	-5.49 (1.13)	35.70 (4.40)	24.38 (2.03)
	Random Search	-7.42 (1.08)	-5.97 (1.22)	35.29 (4.59)	22.19 (1.81)
	Parallel Local Search	**-10.17** (1.43)	**-9.77** (1.49)	39.95 (5.44)	26.96 (2.19)

Table 2: Explanation metrics across experimental settings. 95% confidence intervals are shown in parentheses, obtained by bootstrap with data and model seeds resampled 100k times. Bolded numbers are the best in their group at a statistical significance threshold of $p < .05$.

Models. We train ten seeds of BERT-Base models on each dataset [16], which we term the Standard models (input preprocessing and training hyperparameter details given in Appendix A). Additionally, for each dataset we train another ten Counterfactual-Trained (CT) models using the Attention Mask `Replace` function, following the approach outlined in Sec. 4.

Compute Budget. By default, we use a **budget of 1000 forward and backward passes** (counted together) per explanation. We also consider a special case of a fixed budget per Suff or Comp score, meaning that compute must be allocated across sparsity levels, which favors salience methods over search methods on the basis that one can easily vary the sparsity of explanations with a salience explanation. In this case, we report search methods with a budget of 250 per sparsity level.

7.2 Main Results

Explanation Metrics by Dataset, Method, and Model. In Table 2, we show Suff and Comp scores across methods and datasets, for both the Standard and Counterfactual-Trained (CT) models. We give results for SST-2 later in Table 3 because we use especially short sequences for this dataset, and we

Dataset	Method	Sufficiency ↓		Comprehensiveness ↑	
		Standard Model	CT Model	Standard Model	CT Model
SST-2	LIME	1.98 (0.84)	5.92 (0.93)	52.42 (2.92)	45.75 (2.49)
	Taylor Search	0.09 (0.50)	5.02 (0.79)	45.65 (3.11)	38.91 (2.70)
	Ordered Search	-0.91 (0.47)	2.69 (0.79)	56.24 (2.82)	49.21 (2.48)
	Random Search	-0.91 (0.48)	2.70 (0.79)	56.11 (2.85)	48.98 (2.49)
	Parallel Local Search	-0.91 (0.51)	2.68 (0.85)	56.28 (2.84)	49.25 (2.53)
	Exhaustive Search	-0.91 (0.51)	2.68 (0.85)	56.29 (2.84)	49.26 (2.53)

Table 3: Explanation method performance (including exhaustive search) on short sequences from SST-2. 95% confidence intervals are in parentheses.

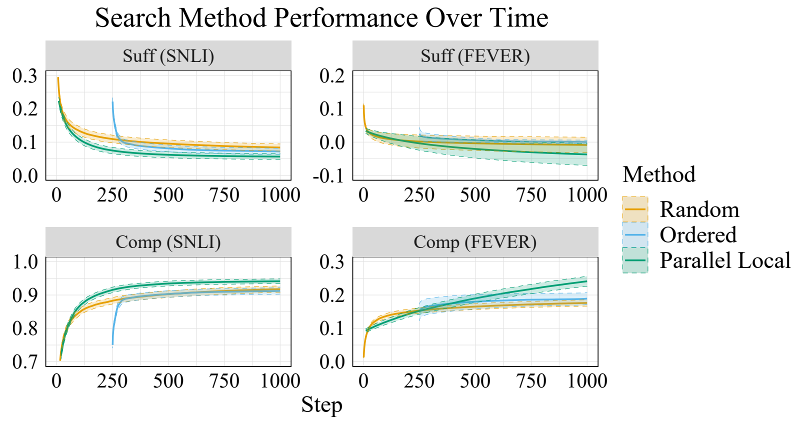

Figure 3: Search method performance over time on FEVER and SNLI with Counterfactual-Trained models, for searches that ran for 1000 forward passes. Shaded areas represent 95% confidence intervals accounting for seed variance using 10 random seeds.

relegate results for Vanilla Gradient to Appendix C as it is universally the worst method we consider. We summarize our key observations as follows:

1. **Parallel Local Search performs the best out of all of the explanation methods, and it is the only method to consistently outperform Random Search**. Improvements in Sufficiency are statistically significant for every dataset using both Standard and CT models, with differences of up to 12.9 points over LIME and 7.6 points over Random Search. For Comprehensiveness, Parallel Local Search is the best method in 9 of 10 comparisons, 7 of which are statistically significant, and improvements are as large as 20.8 points over LIME and 17 points over Random Search.
2. **LIME is the best salience method on both Suff and Comp metrics, but it is still outperformed by Random Search on Sufficiency in 9 of 10 comparisons, by up to 21.5 points**. LIME does appear to perform better than Random Search on Comprehensiveness with three of five datasets for Standard models and four of five with CT models, where the largest improvement over Random Search is 7.49 points.
3. **Suff and Comp scores are often much worse for CT models than for Standard models**. With Random Search, for instance, Comp scores are worse for all datasets (by up to 24.16 points), and Suff scores are worse by 16.92 points for SNLI, though there are not large changes in Suff for other datasets. These differences show that the out-of-distribution nature of counterfactual inputs can heavily influence metric scores, and they lend support to our OOD argument in Sec. 4.

Now we discuss results for SST-2, which are shown in Table 3. Our primary observation here is that most of the search methods we consider perform as well as Exhaustive Search (for sequences short enough to exhaustively search). The closest to Exhaustive Search is Parallel Local Search, which exactly identifies the optimal explanation for the Sufficiency metric and comes within .01 of the optimal Comprehensiveness value. Meanwhile, the best salience method (LIME) underperforms these search methods by between 2.86 and 3.86 points, showing that salience methods fall well behind search methods in this scenario.

Search Method Performance Over Time. In Figure 3, we show search performance across time steps for the three best performing search methods, Random, Ordered, and Parallel Local. Note that Ordered Search begins at step 251 since the first 250 forward passes are allocated to computing LIME, and Parallel Local Search begins at step 10 since we use 10 parallel runs. We see that Parallel Local outperforms Random early on and then continues to remain the preferable method, as differences at step 1000 are statistically significant at $p < .05$. In fact, for FEVER, where the search space is larger, Parallel Local will clearly continue to improve with additional compute, while Random and Ordered plateau in performance by 1000 steps.

Reduced Search Budget. Each of the search methods tends to achieve good performance even by 250 steps. We report results using a reduced search budget of 250 for all datasets in Table 6 in Appendix C. We find that search methods outperform salience methods in 15 of 24 comparisons (with 8 favoring LIME), suggesting that search methods will usually be preferable to salience methods even if one aims to obtain explanations at four sparsity levels with a single compute budget for all four explanations.

Difference in Probabilities vs. Weight of Evidence. Recall that the weight-of-evidence explanation metrics show changes in the (real-valued) model evidence before this evidence is compressed to the bounded probability scale (see Sec. 3). We do not observe any differences in the trends reported here when using the weight-of-evidence versions of Suff and Comp rather than the standard difference-in-probability versions. For a selection of scores from both versions, see Table 8 in Appendix C.

8 Discussion

Should We Prefer Counterfactual-Trained Models If They Are Harder to Explain? This question is raised by the fact that Suff and Comp scores are often worse for CT models (see Sec. 7). We suggest that the only reason for choosing between Standard and CT models is that CT models' explanations are not influenced by the model prior and random seed to the extent that Standard models' explanations are, as we argued in Sec. 4. If one prefers explanations (and explanation metrics) to reflect what a model *has learned from the data*, rather than the model prior and random seed, one would prefer to use CT models. It would be a mistake to prefer the standardly trained models on the grounds that they are "more easily explained" when this difference is due to the way we unintentionally influence model behavior for out-of-distribution data points.

In Defense of Searching For Explanations. De Cao et al. [15] argue against using a certain kind of search to find feature importance explanations on the grounds that it leads to "over-aggressive pruning" of features that a model does in fact rely on for its output. In their case, the objective of the search method is to find "a subset of features (e.g. input tokens) ... [that] can be removed without affecting the model prediction." They suggest that this method is susceptible to *hindsight bias*, asserting that "the fact that a feature can be dropped does not mean that the model 'knows' that it can be dropped and that the feature is not used by the model when processing the example." They provide the example of a model that (successfully) counts whether there are more 8s than 1s in a sequence, where they take issue with the fact that the search method would return a single 8 as its explanation for any sequence with more 8s than 1s, since this preserves the model prediction of "more 8s than 1s." The problem with this explanation, it is said, is that it removes digits that are most certainly being counted by the model when it comes to its decision. They provide empirical evidence that a learned model does in fact count all 8s and 1s before deciding which are more numerous.

One response to this argument is that if one obtains the optimal solution to an optimization problem and is not satisfied with it, then the objective must not be capturing everything that we care about, and the issue is not with the optimization method (i.e. search) that is employed. In the case at hand, we should first note that the objective is actually under-specified. De Cao et al. [15] suppose the search method returns the *maximal set* of tokens that can be removed while maintaining the model prediction, but the objective is not given with any preference for explanation *sparsity* (only that the removed tokens are a "subset" of the input). However, De Cao et al. [15] would take issue with search-based explanations regardless of whether the search method returns the minimal or maximal subset of tokens that can be removed without changing the model prediction. This is because they want the explanation to identify tokens that are "used by the model when processing the example." This criterion is not formalized, but the problem must be that it is a different criterion than the search objective, which is to find a feature subset that preserves the model prediction. After formalizing the notion of a feature being "used by a model," one should then be able to search for good explanations under the corresponding objective.

Explanation Distribution at Train Time. We reiterate that, to exactly match train and test-time distributions, models would be trained on counterfactual inputs drawn from explanation methods themselves, rather than simply random explanations. For now, this remains prohibitively expensive, as it would increase the number of forward passes required during training by up to 1000x depending on the budget to be used when explaining predictions. Future work in reducing the costs of obtaining explanations will help set the stage for more closely aligning the train and test time explanation distributions. There may also be applicable insights to draw from work on efficiently improving model robustness to local perturbations, such as Miyato et al. [40].

Another question that arises during training is whether applying the `Replace` function to x_i implies that the label y_i should change. It may be problematic, for example, to fit a model to (`Replace`(x, e), y) pairs if removing even a small number of tokens in x tends to flip the label. This could be related to the small (around 0.7 percentage point) drops in accuracy observed with training models on counterfactual inputs (see Table 5 in Appendix C). It will be valuable in future work to determine when the `Replace`(x, e) function should be accompanied by certain label transformations for purposes of training CT models.

Measuring Compute Budget. While we choose to use the number of forward and backward passes as our compute budget for each method, wall-clock time will continue to be a useful and practical measure of compute for comparing methods. We note that batching forward passes on a GPU will significantly speed up method runtimes for a single data point while keeping the number of forward passes constant. In our experiments, this means that Parallel Local Search is very efficient compared to Gradient Search, which does not batch inputs in the forward or backward pass. We use forward and backward passes as the unit of our compute budget since this is the fundamentally rate-limiting step in obtaining explanations, but practitioners will do well to compare methods with respect to wall-clock time as well.

9 Conclusion

In this paper, we first provide a new argument for why it is problematic to use out-of-distribution counterfactual inputs when creating and evaluating feature importance explanations. Then, we present our Counterfactual Training solution to the OOD problem, which allows for test-time evaluation of individual explanations. We next compare several common `Replace` functions, finding that Attention Mask and Mask Token functions produce the most in-distribution counterfactuals. However, we find that Counterfactual Training leads counterfactuals to be even more in-distribution, regardless of the `Replace` function, and hence we ultimately recommend the use of Counterfactual Training. After evaluating seven explanation methods across six datasets, we find that our Parallel Local Search method is the best performing method in almost every setting in terms of Sufficiency and Comprehensiveness metrics. We also observe sizable differences in explanation metrics for Standard and CT models, suggesting that our solution to the OOD problem greatly reduces the social misalignment of feature importance explanations.

10 Broader Impacts and Limitations

There are several positive broader impacts of improved feature importance estimation methods and solutions to the OOD problem of the kind discussed here. When model developers and end-users wish to understand the role of certain features in model decisions, FI estimation methods help provide an answer. Access to this kind of information can allow for (1) developers to check that their model is relying on the intended features when it makes decisions, and not unintended features, (2) developers to discover which features are useful to a model accomplishing a task, and (3) users to confirm that a decision was based on features that they deem acceptable, and if not, to pursue some recourse to the decision (provided that recourse is available). Our solution to the OOD problem helps align FI-based explanations with the kind of information that developers and users expect them to convey, e.g. by limiting the influence of the model prior on the explanations.

Nevertheless, there are still some risks associated with the development of FI-based explanations, mainly involving potential misuse and over-reliance. FI-based explanations are *not* summaries of data points or justifications that a given decision was the "right" one. When explanations are good, they reflect what the model has learned, but it need not be the case that what the model has learned is good and worth basing decisions on. It is also important to emphasize that FI-based explanations are not perfect, as there is always possibly some loss of information when limiting what is conveyed to the recipient of an explanation (in our case, making explanations sparse). To the extent that FI-based explanations do not perfectly reflect what the model thinks is important, trust in and acceptance of these explanations should be appropriately calibrated. Lastly, we note that the Counterfactual Training we propose uses an approximation to the test-time counterfactual distribution, and we do not fully eliminate the influence of the random seed on explanations, meaning that FI explanations for the models we consider will not be perfectly socially aligned.

Acknowledgements

We thank Serge Assaad for helpful discussion of the topics here, as well as Xiang Zhou, Prateek Yadav, and Jaemin Cho for feedback on this paper. This work was supported by NSF-CAREER Award 1846185, DARPA Machine-Commonsense (MCS) Grant N66001-19-2-4031, Royster Society PhD Fellowship, Microsoft Investigator Fellowship, and Google and AWS cloud compute awards. The views contained in this article are those of the authors and not of the funding agency.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018. URL https://arxiv.org/pdf/1810.03292.pdf. 8

[2] David Alvarez-Melis, Hal Daumé III, Jennifer Wortman Vaughan, and Hanna Wallach. Weight of evidence as a basis for human-oriented explanations. *arXiv preprint arXiv:1910.13503*, 2019. URL https://arxiv.org/pdf/1910.13503.pdf. 4, 25

[3] Leila Arras, Ahmed Osman, Klaus-Robert Müller, and Wojciech Samek. Evaluating recurrent neural network explanations. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 113–126, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4813. URL https://www.aclweb.org/anthology/W19-4813. 1, 2, 3

[4] Seo-Jin Bang, Pengtao Xie, Wei Wu, and Eric P. Xing. Explaining a black-box using deep variational information bottleneck approach. *ArXiv*, abs/1902.06918, 2019. URL https://arxiv.org/abs/1902.06918. 3, 9

[5] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pages 462–471. PMLR, 2018. URL http://proceedings.mlr.press/v80/baptista18a/baptista18a.pdf. 2, 3, 9, 10, 22

[6] Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *ACL 2019*, pages 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL https://www.aclweb.org/anthology/P19-1284. 3, 9

[7] David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 429–439. PMLR, 2017. URL http://proceedings.mlr.press/v70/belanger17a.html. 3, 9

[8] J Bernardo, J Berger, APAFMS Dawid, A Smith, et al. Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475, 1998. URL http://www.cs.toronto.edu/~radford/ftp/val6gp.pdf. 5

[9] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP 2015*, 2015. URL https://arxiv.org/abs/1508.05326. 6, 10

[10] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. In *ICLR*, 2019. URL https://arxiv.org/pdf/1807.08024.pdf. 3, 6

[11] Hanjie Chen and Yangfeng Ji. Learning variational word masks to improve the interpretability of neural text classifiers. In *EMNLP*, pages 4236–4251, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.347. URL https://www.aclweb.org/anthology/2020.emnlp-main.347. 3, 9

[12] George Chrysostomou and Nikolaos Aletras. Variable instance-level explainability for text classification, 2021. URL https://arxiv.org/pdf/2104.08219.pdf. 3

[13] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015. URL https://arxiv.org/pdf/1502.02127.pdf. 5

[14] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://www.aclweb.org/anthology/N19-1300. 10

[15] Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *EMNLP*, pages 3243–3255, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.262. URL https://www.aclweb.org/anthology/2020.emnlp-main.262. 1, 3, 9, 13

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL 2019*, 2019. URL https://arxiv.org/pdf/1810.04805.pdf. 6, 11

[17] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. Eraser: A benchmark to evaluate rationalized nlp models. In *ACL 2020*, volume abs/1911.03429, 2020. URL https://arxiv.org/pdf/1911.03429.pdf. 1, 2, 3, 4, 10, 23

[18] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*, February 2020. URL http://arxiv.org/abs/2002.06305. arXiv: 2002.06305. 5

[19] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *ACL*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2006. URL https://www.aclweb.org/anthology/P18-2006. 3, 9, 10, 21

[20] Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. CRC press, 1994. 7

[21] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017. URL https://arxiv.org/pdf/1704.03296.pdf. 2, 3, 5, 9

[22] Peter Hase and Mohit Bansal. Evaluating explainable ai: Which algorithmic explanations help users predict model behavior? In *ACL 2020*, 2020. URL https://arxiv.org/pdf/2005.01831.pdf. 1

[23] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1806.10758. 1, 3, 5, 6

[24] Cheng-Yu Hsieh, Chih-Kuan Yeh, Xuanqing Liu, Pradeep Ravikumar, Seungyeon Kim, Sanjiv Kumar, and Cho-Jui Hsieh. Evaluations and methods for explanation through robustness analysis. *arXiv preprint arXiv:2006.00442*, 2020. URL `https://arxiv.org/pdf/2006.00442.pdf`. 3, 5

[25] Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310, 2021. URL `https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00367/98620/Aligning`. 2, 3, 5

[26] Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, 2019. 3

[27] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2016. URL `https://arxiv.org/pdf/1611.01144.pdf`. 9

[28] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pages 2907–2916. PMLR, 2020. URL `https://arxiv.org/pdf/1910.13413.pdf`. 3

[29] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1023. URL `https://www.aclweb.org/anthology/N18-1023`. 10

[30] Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. Interpretation of NLP models through input marginalization. In *EMNLP*, pages 3154–3167, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.255. URL `https://www.aclweb.org/anthology/2020.emnlp-main.255`. 2, 3, 5, 6, 7, 20

[31] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019. URL `https://papers.nips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf`. 3

[32] Eric Lehman, Jay DeYoung, Regina Barzilay, and Byron C. Wallace. Inferring which medical treatments work from reports of clinical trials. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3705–3717, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1371. URL `https://www.aclweb.org/anthology/N19-1371`. 10

[33] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015. URL `https://arxiv.org/pdf/1506.01066.pdf`. 4

[34] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016. URL `https://arxiv.org/pdf/1612.08220.pdf`. 1, 2, 3

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL `https://arxiv.org/pdf/1907.11692.pdf`. 6

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL `https://arxiv.org/pdf/1711.05101.pdf`. 22

[37] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774, 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf. 1

[38] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR 2017*, 2017. URL https://arxiv.org/abs/1611.00712. 9

[39] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019. doi: 10.1016/j.artint.2018.07.007. URL https://doi.org/10.1016/j.artint.2018.07.007. 2

[40] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. URL https://arxiv.org/pdf/1704.03976.pdf. 14

[41] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012. URL https://rtg.cis.upenn.edu/cis700-2019/papers/dataset-shift/dataset-shift-terminology.pdf. 4

[42] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In Mireille Hildebrandt, Carlos Castillo, Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency*, pages 607–617. ACM, 2020. doi: 10.1145/3351095.3372850. URL https://doi.org/10.1145/3351095.3372850. 3, 9

[43] Dong Nguyen. Comparing automatic and human evaluation of local explanations for text classification. In *NAACL-HLT 2018*, 2018. URL https://www.aclweb.org/anthology/N18-1097.pdf. 1, 2, 3

[44] Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. An information bottleneck approach for controlling conciseness in rationale extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1938–1952, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.153. URL https://www.aclweb.org/anthology/2020.emnlp-main.153. 3, 9

[45] Marc Pirlot. General local search methods. *European journal of operational research*, 92(3): 493–511, 1996. 2, 3, 10

[46] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *J. Mach. Learn. Res.*, 20(53):1–32, 2019. URL https://arxiv.org/pdf/1802.09596.pdf. 5

[47] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. URL https://arxiv.org/pdf/1602.04938.pdf. 1, 3, 4, 6, 8

[48] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018. 3

[49] M. Robnik-Sikonja and I. Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20:589–600, 2008. URL http://lkm.fri.uni-lj.si/rmarko/papers/RobnikSikonjaKononenko08-TKDE.pdf. 4, 25

[50] Christian Schäfer. Particle algorithms for optimization on binary spaces. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):1–25, 2013. URL https://arxiv.org/pdf/1111.0574.pdf. 2, 3, 9

[51] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, 2017. 3

[52] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop at International Conference on Learning Representations.*, 2013. 3

[53] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014. URL `https://arxiv.org/abs/1312.6034`. 1, 4

[54] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 3

[55] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1170`. 10

[56] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. URL `https://www.aclweb.org/anthology/D13-1170.pdf`. 6, 23

[57] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020. URL `https://distill.pub/2020/attribution-baselines/`. 3

[58] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR, 2020. URL `https://arxiv.org/pdf/1908.08474.pdf`. 3

[59] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328, 2017. 3

[60] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, June 2017. URL `http://arxiv.org/abs/1703.01365`. 1, 2, 3, 4, 5, 6, 8, 9

[61] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL `https://www.aclweb.org/anthology/N18-1074`. 10

[62] Richard E Turner. Gaussian processes: From the basics to the state-of-the-art, 2016. URL `http://cbl.eng.cam.ac.uk/pub/Public/Turner/News/imperial-gp-tutorial.pdf`. 5, 24

[63] Jan N Van Rijn and Frank Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018. URL `https://arxiv.org/pdf/1710.04725.pdf`. 5

[64] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, 2019. 3

[65] Maksymilian Wojtas and Ke Chen. Feature importance ranking for deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5105–5114. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/36ac8e558ac7690b6f44e2cb5ef93322-Paper.pdf. 3, 9

[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910, 2019. 23

[67] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6. 22

[68] Jihun Yi, Eunji Kim, Siwon Kim, and Sungroh Yoon. Information-theoretic visual explanation for black-box classifiers. *arXiv preprint arXiv:2009.11150*, 2020. URL https://arxiv.org/pdf/2009.11150.pdf. 2, 3, 5, 6, 7, 20

[69] Fan Yin, Zhouxing Shi, Cho-Jui Hsieh, and Kai-Wei Chang. On the faithfulness measurements for model interpretations, 2021. URL https://arxiv.org/pdf/2104.08782.pdf. 3

[70] Omar Zaidan, Jason Eisner, and Christine Piatko. Using "Annotator Rationales" to Improve Machine Learning for Text Categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York, April 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N07-1033. 2, 3, 5

[71] Ruiqi Zhong, Steven Shao, and Kathleen McKeown. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*, 2019. URL https://arxiv.org/pdf/1908.06870.pdf. 3

[72] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? *arXiv preprint arXiv:2104.14403*, 2021. URL https://arxiv.org/pdf/2104.14403.pdf. 1

[73] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR*, 2017. URL https://arxiv.org/pdf/1702.04595.pdf. 2, 3, 6

A Method Implementation and Hyperparameter Tuning Details

A.1 `Replace` Functions

1. **Attention Mask.** Described in main paper.
2. **Marginalize.** In marginalization, the final prediction is obtained from the marginal log probability as $\arg\max_y \ln \sum_{\tilde{x}\sim p_\phi(\tilde{x}|x,e)} p_\theta(y|\tilde{x})p_\phi(\tilde{x}|x,e)$, where $p_\phi(\tilde{x}|x,e)$ is the distribution over imputed tokens. Since computing this marginal distribution is quite expensive, we adopt a Monte Carlo approximation common to past work [30, 68]. Using a subset of SNLI validation data, we tune the number of samples over sizes in {10, 25, 50, 100}, selecting for maximum robustness. Surprisingly, 10 samples performed the best in terms of robustness, though the margin was small over the other values. Consequently, we select a value of 10, which also allows us to evaluate this method at scale due to its relative computational efficiency. This finding is similar to the results in Yi et al. [68], who ultimately use a value of 8 samples for Monte Carlo estimation of the marginal distribution. This method is still over ten times slower than other `Replace` functions given the need to perform many MLM forward passes.

3. **MASK Token**. Described in main paper.
4. **Slice Out**. Desribed in main paper.
5. **Zero Vector**. Described in main paper.

A.2 Explanation Methods

LIME. Here, we specify the form of the weight function π, the regularization method Ω, and the distribution of perturbed data points $p(\tilde{x}|x_i)$, which are all set to the default LIME package settings. The weight function π is an exponential kernel on the negative cosine similarity between data points multiplied by 100. The perturbation distribution is over binary vectors: in every sample, a uniformly random number of randomly located elements are set to 0, and the remainder are kept as 1. Lastly, Ω is to perform forward selection when there are no more than 6 features (i.e. perform greedy local search in the space of possible feature sets, starting with no features and adding one feature at a time). When there are more than six features, ridge regression is used, then the top k features according to the product of their feature weight and the observed feature value (0 or 1 in our case). We use the regression weights as the final salience scores.

Vanilla Gradient. See main paper for description.

Integrated Gradients. See main paper for description.

Random Search. Using a subset of SNLI validation points, we tune this method over two possible search spaces: the space of all k-sparse explanations, when the sparsity levels allows up to k tokens to be retained (or no lower than k tokens, for Comprehensiveness), and the space of all allowably sparse explanations. We find it preferable to restrict the search space to exactly k-sparse explanations. We adopt this same search space for all other search methods.

Gradient Search. Here, we give more details to checkpoint selection, weight initialization, regularization, and tuning for Gradient Search. For checkpoint selection: we select the search state that achieves the best Sufficiency (or Comprehensiveness) as measured once every m gradient updates. We do so because checking these metrics consumes some of the available compute budget (see Appendix B.2), and therefore we check the metric value at intervals for purposes of checkpointing. In our experiments, we check the metric every 20 gradient updates and search until the total budget has been consumed. For initialization: a random initial starting point is sampled from a Multivariate Normal distribution centered around 0, with $\Sigma = I$. For regularization and other tuning details, we perform sequential line searches over hyperparameters, according to Sufficiency scores on a subset of BoolQ data points. To tune a specific hyperparameter, we set all other hyperparameters to some default values. We refer to the hyperparameters we use after tuning as "final" hyperparameters, which are listed in the table below (note: Number of Samples is the number of sampled explanations per gradient update).

Hyperparameter	Default	Final	Range
Number of Samples	10	1	1, 10, 20, 40
Optimizer	AdamW	AdamW	AdamW, SGD
Scheduler	None	None	None, Linear, Step, Cosine
Learning Rate	0.2	0.1	0.01, 0.05, 0.1, 0.2, 0.4
Sparsity Weight	1e-3	1e-3	1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 0
Target Sparsity	0.1	0.05	0.03, 0.05, 0.1, 0.2, 0.3, 0.4

Taylor Search. Taylor search uses a Taylor expansion of the model loss to forecast changes in the loss from taking steps in the search space. Following [19], we employ this search heuristic in a beam search. Hyperparameters for Taylor Search are listed below. We performed tuning with Taylor Search for Sufficiency on a subset of BoolQ validation points, and ultimately we selected the largest, best performing pair of values given the available compute budget.

Hyperparameter	Default	Final	Range
Beam Width	2	5	1,2,3,4,5
Number of Steps	50	50	50, 100, 200

Ordered Search. Using Sufficiency scores on a subset of SNLI validation data, we tune over the ratio between compute budget used in estimating the model f_θ (i.e. salience scores) and the budget used for the search. Out of 1000 steps, we consider using up to m steps for estimating the salience scores via LIME, where $m \in \{10, 100, 200, 250, 500, 750\}$, ultimately using $m = 250$.

Parallel Local Search. Here we specify the Propose function used in Parallel Local Search, and we describe some additional implementation details, some comparisons we performed with Simulated Annealing, and tuning for the number of parallel searches r. The Propose function samples new explanations by starting a random walk from the current explanation that ends when a not-before-seen explanation is encountered. As in Taylor Search, neighboring states have Hamming distance 2. Though we parallelize this search method, we maintain a shared set of previously-seen explanations and compute the Propose function serially at each step so that we never compute the more expensive objective function on the same explanations.

A similar algorithm, Simulated Annealing, uses a probabilistic update condition that favors exploration early on in the search and exploitation later in the search [5]. We find it preferable to use a deterministic update rule, moving to the new state if and only if its objective value is better than the old state. Lastly, following tuning results, we use $r = 10$ parallel runs for all experiments, meaning that each run has a budget $b = 100$ when the overall method budget is 1000 forward passes. The value of r is tuned over the set {1, 5, 10, 25}. We note that using a value greater than 1 significantly improves the wall-clock runtime of this algorithm, as the batched forward passes performed when r searches are done in parallel are much more efficient than performing a greater number of forward passes with only 1 input.

A.3 Model Training Details and Experiment Runtimes

We now give implementation details for training models on our six datasets. The models include BERT-Base or RoBERTa-Base models drawn from the Hugging Face Transformers library [67], trained with AdamW [36] using a learning rate of 1e-5, and in general we select the best model training checkpoint according to the validation set accuracy. When training models for our analysis of counterfactual OOD-ness in Sec. 5, we train Standard models for 20 epochs and Counterfactual-Trained models for 10 epochs, since in the latter case we effectively double the number of inputs per batch. For our explanation evaluation experiments in Sec. 7, we train all models for 10 epochs. Note that in every experiment we train ten models using ten different random seeds.

All experiments are conducted on NVIDIA RTX 2080 GPUs. Wall-clock runtimes for training models are: for Sec. 5 experiments, .6 hours per SNLI model and 1 hour per SST-2 model; for Sec. 7, training one model takes 4.8 hours for FEVER, .7 hours for BoolQ, 1.5 hours for SNLI, 2.9 hours for MultiRC, 1.7 for Evidence Inference, and 3.9 hours for SST-2.

For analysis and explanation evaluation experiments, we report the following runtimes: Sec. 5 experiments take up to 12 hours for robustness analysis for each `Replace` function (across seeds, models, and datasets), except for Marginalize which takes close to 48 hours. We give max runtimes across datasets for obtaining and evaluating explanations to provide an upper bound on wall-clock runtime given the variable sequence lengths between datasets, meaning we report runtimes for the Evidence Inference dataset. With a compute budget of 1000 forward/backward passes, we find observe that obtaining a set of explanations at one sparsity level for 500 data points using one BERT-Base model takes roughly 3 hours for LIME, 3 minutes for Vanilla Gradient, 10 hours for Integrated Gradients, 10 hours for Gradient Search, 10 hours for Taylor Search, 3 hours for Ordered Search, 3 hours for Random Search, and 3 hours for Parallel Local Search.

Dataset	# Classes	Split	Size	Avg. document length	Avg. query length
SNLI	3	Train	5000	24.8	-
		Validation	9823	24.4	-
		Test	9807	24.4	-
BoolQ	2	Train	9427	121.9	9.4
		Validation	3270	119.8	9.3
FEVER	2	Train	97957	342.3	10.4
		Validation	6122	291.2	10.7
		Test	6111	278.7	10.7
Evidence Inference	3	Train	7958	483.1	25.3
		Validation	972	484.4	23.6
		Test	959	482.0	27.0
SST-2 2	2	Train	67349	11.3	-
		Validation	872	23.2	-
		Test	1821	10.8*	-
MultiRC 2	2	Train	24029	326.9	21.2
		Validation	3214	326.1	20.7
		Test	4848	314.8	20.8

Table 4: Dataset statistics, including the number of points and the average number of tokens per document (and query) for each dataset, as per the BERT-Base tokenizer. *For SST-2, we use the shortest 500 test points.

B Experimental Details

B.1 Data Preprocessing

We use datasets as prepared by the ERASER dataset suite [17], which are publicly available and distributed under the original licenses for each dataset (including Creative Commons and MIT License), as well as SST-2 which is publicly available under a Creative Commons license [56]. Note that BoolQ has only one split for evaluation. Additionally, note that for experiments in Sec. 7, we use a subset of 50k points for training models on SNLI, and we use the 500 shortest SST-2 test points according to the number of tokens given by the BERT tokenizer [66].

For preprocessing, input text is split by spaces into a list of words. We tokenize each word independently and concatenate the resulting tokens. Each input consists of a document section and a query section. The document section contains the document while the query section contains the question. The two sections are separated by [SEP]. The entire input is preceded by a [CLS] token and followed by a [SEP] token. Note that these special tokens are never replaced by the `Replace` function. For inputs longer than 512 tokens (the maximum input length for our task model Bert), we truncate the input document so that the entire input is shorter or equal to 512 tokens.

B.2 Compute Budget Details

In this section we describe how the compute budget is spent by each explanation method. Note that we consider both creating and evaluating explanations to draw from the available compute budget, because some methods compute the Suff and Comp metrics while obtaining an explanation, whereas others leave these metrics to be checked after an explanation is settled on. We describe the standard case in this paper of 1000 forward and backward passes per final metric value.

1. LIME uses 996 forward passes to obtain an explanation, then 4 forward passes to obtain a final metric value (one per sparsity level).
2. Vanilla Gradient uses only a single forward and backward passes. This is our only method that uses a fixed compute budget.

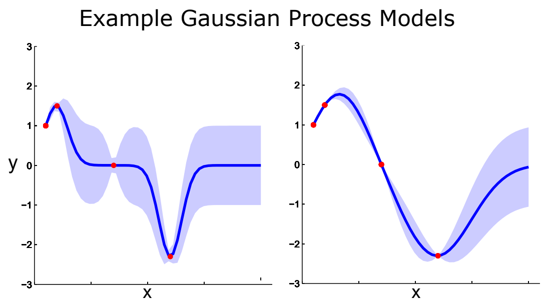

Figure 4: A visual aid to our argument in Sec. 4 (figure adapted from Turner [62]). Predictions from Gaussian Process models revert back to their mean function, in this case 0, in regions without data. The model on the right reverts back to the mean more slowly due to the choice of covariance function.

Dataset	Standard Acc.	CT Acc.
SNLI	85.84 (0.69)	85.08 (0.71)
BoolQ	74.16 (1.62)	73.76 (1.62)
FEVER	89.66 (0.76)	89.72 (0.76)
Evidence Inference	58.81 (3.12)	57.35 (3.13)
SST-2	92.89 (1.18)	92.43 (1.21)
MultiRC	68.96 (1.30)	67.76 (1.32)

Table 5: Accuracies for Standard models and CT models.

3. Integrated Gradients uses 498 forward and backward passes and 4 forward passes to obtain the final metric value.
4. Taylor Search uses no more than 1000 forward passes, given the beam width and number of steps described in Appendix A.
5. The remaining search methods (including Ordered Search, Random Search, Parallel Local Search) all use 1000 forward passes in total, since these methods involve exactly computing the objective value at each step, so the metrics do not need to be recomputed after explanations are obtained.

C Additional Results

Counterfactual-Trained Model Accuracy. In Table 5, we show model accuracies on test sets for standardly trained models (Standard) and Counterfactual-Trained (CT) models, with 95% confidence intervals reflecting sample variance in parentheses. These accuracies are from the single best performing seed for each dataset, according to dev set accuracies, out of 10 trained models. Note that for SST-2, we report dev set numbers as test labels are unavailable.

We observe that differences in accuracies are typically less than 1 point between the two training algorithms. On average across the datasets, the difference is about 0.7 points. This is a small but consistent gap, and hence it will be valuable for future work to study the causes of this gap and identify ways to align the train and explanation-time distributions without losing any model accuracy on test data.

Results with Reduced Search Budget. In Table 6, we give results for a reduced search budget of 1000 forward passes *across* sparsity levels, i.e. 250 per sparsity level. This setup favors salience-based methods which can easily give explanations at varying sparsity levels. With too many sparsity levels, **this evaluation is heavily biased toward salience-based methods**, since it spreads the compute for search methods across sparsity levels. We use a constant budget per sparsity for results in the main paper because we view the use of multiple sparsity levels as an attempt to average results across possible desired settings, and explanations at multiple sparsities may not always be needed. But ultimately, user preferences will dictate whether explanations at multiple levels of sparsity are desired. This discussion aside, the results are as follows: compared to the best salience-based method, LIME, **search methods are preferable in 15 of 24 comparisons**, while **LIME is preferable in 8 of 24**

		Sufficiency ↓		Comprehensiveness ↑	
Dataset	Method	Standard Model	CT Model	Standard Model	CT Model
SNLI	Vanilla Grad	59.41 (2.42)	63.41 (0.81)	7.08 (1.63)	5.84 (1.06)
	LIME	20.00 (2.02)	27.09 (1.68)	82.17 (2.82)	75.34 (1.94)
	Ordered Search (250)	-1.19 (0.87)	16.23 (1.45)	87.01 (2.40)	83.31 (1.73)
	Random Search (250)	-1.34 (0.90)	16.82 (1.46)	86.10 (2.52)	82.45 (1.82)
	PL Search (250)	**-1.50** (0.96)	**15.30** (1.37)	**87.25** (2.42)	**84.57** (1.68)
BoolQ	Vanilla Grad	30.81 (3.44)	16.40 (2.13)	2.43 (0.70)	2.25 (0.73)
	LIME	2.14 (1.75)	-1.56 (0.64)	52.03 (3.67)	**36.26** (3.44)
	Ordered Search (250)	1.44 (1.29)	-1.71 (0.70)	43.33 (3.18)	27.78 (3.00)
	Random Search (250)	0.09 (0.82)	-1.84 (0.66)	49.46 (3.55)	27.58 (2.74)
	PL Search (250)	**-0.56** (0.57)	**-2.61** (0.68)	**54.85** (3.78)	31.98 (2.97)
Evidence Inference	Vanilla Grad	20.76 (4.14)	12.96 (2.13)	2.92 (1.31)	1.57 (0.56)
	LIME	-16.07 (2.84)	-14.93 (1.38)	**47.61** (5.66)	**33.97** (4.19)
	Ordered Search (250)	-14.47 (2.65)	-11.67 (1.34)	39.69 (4.83)	26.51 (3.15)
	Random Search (250)	-15.28 (2.67)	-10.86 (1.28)	38.79 (5.46)	23.65 (2.80)
	PL Search (250)	-16.18 (3.14)	-13.78 (2.53)	41.90 (8.96)	25.27 (2.78)
FEVER	Vanilla Grad	19.63 (2.39)	13.21 (1.81)	1.52 (0.60)	1.02 (0.41)
	LIME	-0.24 (0.50)	1.36 (2.13)	**33.86** (3.44)	**22.06** (4.10)
	Ordered Search (250)	-0.73 (0.40)	1.00 (0.90)	28.70 (3.18)	16.30 (2.26)
	Random Search (250)	-1.16 (0.50)	-0.30 (2.07)	29.13 (3.18)	16.58 (1.91)
	PL Search (250)	-1.06 (0.44)	-0.36 (2.04)	25.81 (2.87)	15.22 (1.84)
MultiRC	Vanilla Grad	21.16 (3.47)	11.80 (1.38)	3.75 (1.18)	1.74 (0.79)
	LIME	-5.20 (1.19)	**-5.91** (1.19)	**39.75** (4.80)	**28.57** (2.18)
	Ordered Search (250)	-5.02 (1.03)	-4.16 (1.10)	33.26 (4.25)	21.95 (1.92)
	Random Search (250)	**-6.08** (1.17)	-4.86 (1.20)	32.31 (4.30)	19.95 (1.67)
	PL Search (250)	-5.20 (1.30)	-4.91 (1.27)	28.38 (3.85)	17.46 (1.49)
SST-2	Vanilla Grad	47.26 (3.79)	46.83 (1.41)	2.56 (0.71)	3.01 (1.00)
	LIME	1.97 (0.84)	5.92 (0.93)	52.42 (2.93)	45.74 (2.52)
	Ordered Search (250)	-0.91 (0.47)	2.71 (0.79)	55.90 (2.84)	48.96 (2.50)
	Random Search (250)	-0.91 (0.46)	2.73 (0.75)	55.44 (2.52)	48.31 (2.18)
	PL Search (250)	-0.91 (0.47)	2.69 (0.79)	56.14 (2.84)	49.08 (2.50)

Table 6: Comparing salience methods against search methods with a reduced budget of 1000 forward passes across sparsity levels (250 per sparsity), which favors salience methods. 95% confidence intervals are shown in parentheses, obtained by bootstrap with data and model seeds resampled 100k times. Bolded numbers are the best in their group at a statistical significance threshold of $p < .05$.

comparisons (at statistical significance of $p < .05$). Comparing within search methods, Parallel Local Search is preferable to Random Search 7 times and Random Search is preferable one time (at $p < .05$). We observe that LIME performs best on Comprehensiveness, and therefore we suggest that LIME may be the best method when explanations are desired at many sparsity levels, the compute budget is heavily limited, and one is optimizing for Comprehensiveness. Otherwise, Parallel Local Search remains the preferable method.

Results with RoBERTa Models. Shown in Table 7, we give a comparison between LIME, Random Search, and Parallel Local Search using RoBERTa-Base as our task model, as opposed to the BERT-Base setting in the main paper experiments. Results are given for a subset of datasets, with the same compute budget as in the main paper, using 10 Counterfactual-Trained RoBERTa-Base models on each dataset. We observe similar trends and improvements using Parallel Local Search as with BERT-Base. Parallel Local Search is the best method in each condition and consistently outperforms Random Search, unlike LIME.

Weight of Evidence Metrics. Lastly, in Table 8, we compare the default difference-in-probability version of our metric (reported in the main paper) with the weight-of-evidence version, which is used by [49, 2]. We do not observe any notable differences in the trends between the metrics. We report one case where a hypothesis test is statistically significant with WoE but not the difference in probabilities; however, the difference between p-values is negligible ($p = .052$ vs. $p = .030$).

Dataset	Method	Sufficiency ↓	Comprehensiveness ↑
SNLI	LIME	24.42 (1.65)	71.30 (2.63)
	Random Search	11.52 (1.46)	81.43 (2.59)
	Parallel Local Search	**9.73** (1.41)	**83.44** (2.31)
FEVER	LIME	1.22 (0.81)	25.09 (2.56)
	Random Search	1.10 (0.78)	22.46 (2.50)
	Parallel Local Search	**-0.40** (0.63)	**27.61** (2.68)

Table 7: Explanation metrics for RoBERTa-Base models (Counterfactual-Trained only). 95% confidence intervals are shown in parentheses, obtained by bootstrap with data and model seeds resampled 100k times. Bolded numbers are the best in their group at a statistical significance threshold of $p < .05$.

		Sufficiency ↓		Comprehensiveness ↑	
Dataset	Method	Diff. in Probs	WoE	Diff. in Probs	WoE
Evidence Inference	LIME	-14.93 (1.38)	-0.98 (0.14)	33.97 (4.17)	1.76 (0.29)
	Random Search	-12.71 (1.29)	-0.81 (0.13)	26.50 (3.15)	1.35 (0.21)
	Parallel Local Search	**-20.33** (2.46)	**-1.44** (0.14)	38.71 (3.52)	**2.09** (0.25)
MultiRC	LIME	-5.90 (1.19)	-0.33 (0.09)	**28.57** (2.18)	**1.54** (0.17)
	Random Search	-6.58 (1.20)	-0.39 (0.09)	22.79 (1.78)	1.24 (0.13)
	Parallel Local Search	**-9.77** (1.45)	**-0.63** (0.13)	26.96 (2.05)	1.45 (0.16)

Table 8: Comparison of difference-in-probability and weight-of-evidence versions of Sufficiency and Comprehensiveness, for Counterfactual-Trained models. Trends in results are effectively the same for both versions of the metric. The one difference in hypothesis tests is for Comprehensiveness on Evidence Inference, but the difference in p-values is negligible ($p = .052$ vs. $p = .030$).