GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction

Thai Le
The Pennsylvania State University
thaile@psu.edu

Suhang Wang
The Pennsylvania State University
szw494@psu.edu

Dongwon Lee
The Pennsylvania State University
dongwon@psu.edu

Abstract

Despite the recent development in the topic of explainable AI/ML for image and text data, the majority of current solutions are not suitable to explain the prediction of neural network models when the datasets are tabular and their features are in high-dimensional vectorized formats. To mitigate this limitation, therefore, we borrow two notable ideas (i.e., "explanation by intervention" from causality and "explanation are contrastive" from philosophy) and propose a novel solution, named as GRACE, that better explains neural network models' predictions for tabular datasets. In particular, given a model's prediction as label X, GRACE intervenes and generates a minimally-modified contrastive sample to be classified as Y, with an intuitive textual explanation, answering the question of "Why X rather than Y?" We carry out comprehensive experiments using eleven public datasets of different scales and domains (e.g., # of features ranges from 5 to 216) and compare GRACE with competing baselines on different measures: *fidelity*, *conciseness*, *info-gain*, and *influence*. The user-studies show that our generated explanation is not only more intuitive and easy-to-understand but also facilitates end-users to make as much as 60% more accurate post-explanation decisions than that of LIME. We release the source code of GRACE at https://github.com/lethaiq/GRACE_KDD20

Keywords

explainability, contrastive, data generation

1 INTRODUCTION

Tabular data is one of the most commonly used data formats. Even though tabular data receives far less attention than computer vision and NLP data in neural networks literature, recent efforts (e.g., [1, 2, 20, 27]) have shown that neural networks, deep learning in particular, can also achieve superior performance on this

Feat	freq_now	freq_credit	freq_!!!	freq_!	class
x_1	0.1	0.0	0.0	0.0	Ham
$\tilde{x}_1$	0.1	0.0	**0.3**	**0.453**	**Spam**

Feat	freq_you	freq_direct	avg_longest_capital	class
x_2	0.68	0.34	158.0	Spam
$\tilde{x}_2$	0.68	0.34	**1.0**	**Ham**

Table 1: Examples of original samples x_i and contrastive samples $\tilde{x}_i$ on *spam* dataset. $\tilde{x}_i$ only differs from x_i on *a few* features. (unchanged features are randomly selected)

type of data. Yet, there is still a lack of interpretability that results in the distrust of neural networks trained on general tabular data domains. This obstructs the wide adoption of such models in many high-stakes scenarios in which tabular data is prominent–e.g., healthcare [3, 24], finance [6, 8], social science [14, 37], and cybersecurity [18, 33]. Moreover, the majority of explanation algorithms (e.g., [4, 11, 16, 23, 25, 29, 35]) are designed for models trained on images or texts, while insufficient efforts have been made to explain the prediction results of neural models that take tabular data formats as input. Furthermore, most of the previous explanation approaches are geared for professional users such as ML researchers and developers rather than lay users and ML consumers. This situation calls for a novel approach to provide end-users with the intuitive explanation of neural networks trained on tabular data. However, developing such an approach poses several challenges.

Challenges. *First*, tabular data used in neural network models sometimes have high-dimensional inter-correlated features. Therefore, presenting feature importance scores for top-k or all features (e.g., [25]) can induce both information overload and redundancy, causing confusion to end-users. In fact, for a data instance, a complex model can focus on just a few key features in making its prediction. To illustrate, Table 1 shows that for two emails x_1 and x_2, the model can focus on two different sets of features, $freq\_!$, $freq\_!!!$ or $avg\_longest\_capital$, respectively, to predict if an email is a spam or ham. While explanation constructed **only** from these features is much more concise, providing both $freq\_!$ and $freq\_!!!$ (frequency of "!" and "!!!" within an email content) in the first example produces redundancy. In this case, we also want to replace $freq\_!$ with another key feature to make the explanation more informative. Thus, we need to find a subset of instance-dependent key features that are both *concise* and *informative* to explain the model's prediction.

Second, for images or texts, highlighting a patch of an image (e.g., [4, 16, 35]) or a phrase of a sentence (e.g., [11]) usually gives a clear understanding of what a model is focusing on and why a model gives such prediction. However, in tabular data, such visualization does not provide much insight into the chosen model. For instance, in the second example in Table 1, the model predicts x_2 as spam

and the important feature used by the model is *avg_longest_capital*. However, simply providing this feature to end-users does not give an easy-to-understand explanation. Since we often justify our decision verbally [17], in this case, an explanation written in text can help end-users understand the prediction better.

Third, approximating the decision boundaries does not necessarily provide a clear understanding on the decision-making of a model to end-users, who usually lack ML background. Instead, such lay users are usually more interested in the **contrastive explanation**, i.e., why X rather than Y. For example, Table 1 shows in the second example that "had *avg_longest_capital* (i.e., the average length of the longest capitalized words) been about 150 characters shorter, the email would have been classified as **ham rather than spam**". Hence, we need to come up with a new explanation model such as *contrastive explanation* to better explain a model's prediction to lay end-users.

Overview. To sum up, the effort towards generating an explanation that is easy for end-users to understand is challenging, yet also in great demand. Therefore, we propose a novel algorithm, GRACE (*GeneRAting Constrastive samplEs*), which generates and provides end-users with intuitive and informative explanations for neural networks trained on general tabular data. Inspired from Database (DB) literature [21, 26, 32], GRACE borrows the idea of "explanation by intervention" from causality [15, 28] to come up with contrastive explanation–i.e., why a prediction is classified as X rather than Y. Specifically, for each prediction instance, GRACE generates an explainable sample and its contrastive label by selecting and modifying a few instance-dependent key features under both *fidelity*, *conciseness* and *informativeness* constraints. Then, GRACE aims to provide a friendly text explanation of *why X rather than Y* based on the newly generated sample. The main contributions of the paper are:

- We introduce an explanation concept for ML by marrying "contrastive explanation" and "explanation by intervention", then extend it to a novel problem of generating contrastive sample to explain why a neural network model predicts X rather than Y for data instances of tabular format;
- We develop a novel framework, GRACE, which finds key features of a sample, generates contrastive sample based on these features, and provides an explanation text on why the given model predicts X rather than Y with the generated sample; and
- We conduct extensive experiments using eleven real-world datasets to demonstrate the quality of generated contrastive samples and the effectiveness of the final explanation. Our user-studies show that our generated explanation texts are more intuitive and easy-to-understand, and enables lay users to make as much as 60% more accurate post-explanation decisions than that of LIME.

2 THE EXPLANATION MODEL

2.1 Contrastive Explanation

Understanding the answer to the question "Why?" is crucial in many practical settings, e.g., in determining why a patient is diagnosed as benign, why a banking customer should be approved for a housing loan, etc. The answers to these "Why?" questions can be really answered by studying causality, which depicts the relationship between an event and an outcome. The event is a cause if the outcome is the consequence of the event ([21, 28]). However, causality can only be established under a controlled environment, in which one alters a single input while keeping others constant, and observes the change of the output. Bringing causality into data-based studies such as DB or ML is a very challenging task since causality cannot be achieved by using data alone ([21]). As the first step to understand causality in data-intensive applications, DB and ML researchers have tried to lower the bar of explanation, aiming to find the subset of variables that are best correlated with the output. Specifically, DB literature aims to provide explanations for a complex query's outputs given all tuples stored in a database, while ML researcher is keen on explaining the predictions of learned, complex models.

2.2 Explanation by Intervention

By borrowing the notion of *intervention* from causality literature, in particular, DB researchers have come up with a practical way of explaining the results of a database query by searching for an explainable predicate $\mathcal{P}$. Specifically, $\mathcal{P}$ is an explanation of outputs $\mathbf{X}$ if the removal of tuples satisfying predicate $\mathcal{P}$ also changes $\mathbf{X}$ while keeping other tuples unchanged ([21, 26, 32]). Similarly, by utilizing the same perspective, we want to formulate a definition of *explanation by intervention* for ML models at instance-level as follows.

DEFINITION 1 (CONTRASTIVE EXPLANATION (IN ML) BY INTERVENTION). *A predicate $\mathcal{P}$ of subset of features is an explanation of a prediction outcome $\mathbf{X}$, if changes of features satisfying the predicate $\mathcal{P}$ also changes the prediction outcome to $\mathbf{Y}(\neq \mathbf{X})$, **while keeping other features unchanged**.*

For example, possible predicates to explain a spam detector are shown in Table 1. Particularly, predicate $\mathcal{P}_2$: "avg_longest_capital = 1.0" explains why sample $\boldsymbol{x}_2$ is classified as spam rather than ham. Given a prediction of a neural network model on an input, there will be possibly many predicates $\mathcal{P}$ satisfying Def. 1. Hence, it is necessary to have a measure to describe and compare how much influence predicate(s) $\mathcal{P}$ have on the final explanation. Following the related literature of explanation from the DB domain [32], we also formally define a scoring function $infl_{\lambda}(\mathcal{P})$ as the measure on the influence of $\mathcal{P}$ on the explanation with a tolerance level λ as.

DEFINITION 2 (INFLUENCE SCORING FUNCTION).
$$infl_{\lambda}(\mathcal{P}) = \frac{\mathbb{1}(\mathbf{Y} \neq \mathbf{X})}{(Number\ of\ features\ in\ \mathcal{P})^{\lambda}} \quad (1)$$
where $\mathbb{1}(\cdot)$ is an indicator function, $\mathbf{X}$ and $\mathbf{Y}$ are predicted labels before and after intervention, respectively.

The larger the score is, the more influential $\mathcal{P}$ has on the explanation. Hence, $\lambda = 0$ would imply infinite tolerance on the number of features in $\mathcal{P}$, $\lambda > 0$ would prefer a small size of $\mathcal{P}$ and $\lambda < 0$ would prefer a large size of $\mathcal{P}$. In practice, $\lambda > 0$ is preferable because a predicate $\mathcal{P}$ containing too many features would adversely affect the comprehension of the explanation. For example, $infl_1(\mathcal{P}_2) = 1.0$

2.3 From Intervention to Generation

Searching for $\mathcal{P}$ is a non-trivial problem. From Def. 1, we want to approach this problem from a generation perspective. Particularly,

given an arbitrary sample classified as X by a neural network, we want to intervene and modify a small subset of its features to generate a new sample that crosses the decision boundary of the model to class Y. This subset of features and their new values will result in a predicate $\mathcal{P}$. This newly generated sample will help answer the question "Why X rather than Y?". To illustrate, Table 1 shows that $\tilde{x}_2$ is generated samples that correspond to predicate $\mathcal{P}_2$: "avg_longest_capital = 1.0". Using $\mathcal{P}_2$, we can generate an explanation text to present to the users such as "Had the average length of the longest capitalized words been 1.0, the message would have been classified as **ham rather than spam**".

3 OBJECTIVE FUNCTION

Let $f(\cdot)$ be a neural network model that we aim to give instance-level explanation. Denote $X \in \mathbb{R}^{N \times M} = \{x_1, x_2, ..x_n\}, \mathcal{Y} = \{y_1, y_2, ..y_n\}$ as the features and ground-truth labels of data on which $f(\boldsymbol{x})$ is trained, where N, M is the number of samples and features, respectively. X^i and X^j are the i-th and j-th feature, respectively, in features set X. $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ are the i-th and j-th feature of $\boldsymbol{x}$, respectively. First, we want to generate samples that are contrastive. We define such characteristic as follows.

DEFINITION 3 (CONTRASTIVE SAMPLE). *Given an arbitrary $\boldsymbol{x} \in X$, $X \in \mathbb{R}^{N \times M}$ and neural network model $f(\cdot)$, $\tilde{x}$ is called contrastive or contrastive sample of $\boldsymbol{x}$ when:*

$$\min_{\tilde{\boldsymbol{x}}} dist(\boldsymbol{x}, \tilde{x}) \quad s.t. \quad argmax(f(\boldsymbol{x})) \neq argmax(f(\tilde{\boldsymbol{x}})) \quad (2)$$

Then, formally, we study the following problem:

PROBLEM: Given $\boldsymbol{x}$ and neural network model $f(\cdot)$, our goal is to generate new contrastive sample $\tilde{x}$ to provide concise and informative explanation for the prediction $f(\boldsymbol{x})$.

Existing works on adversarial example generation [19, 22] usually define $dist(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ as $\|\boldsymbol{x} - \tilde{x}\|_2^2$, which allows all features to be changed to generate $\tilde{\boldsymbol{x}}$. Though such approaches can generate realistic labeled contrastive samples, they are not appropriate for generating instance-level explanation that are easy to understand because all features are changed.

Instead, we aim to generate $\tilde{x}$ labeled $\tilde{y}$ such that it is minimally different from original input $\boldsymbol{x}$ in terms of only a *few important features* instead of *all features*. Specifically, we desire to explain "Why X rather than Y?" by presenting a *concise* explanation in which only a few features are corrected, e.g., the first example of Table 1 shows if only frequency of '!' is increased to 45.3%, *while keeping other features unchanged*, the email will be classified as "Spam" rather than "Ham". Hence, the less the number of features need to change from $\boldsymbol{x}$ to generate $\tilde{x}$, the more "concise" the explanation becomes. To achieve this goal, we add the constraint as

$$|\mathcal{S}| \leq K \quad (3)$$

where $\mathcal{S}$ is the feature set of $\boldsymbol{x}$ that are perturbed to generate $\tilde{\boldsymbol{x}}$, hence making $|\mathcal{S}|$ the number of features changed, i.e.,

$$|\mathcal{S}| = \sum_{m=1}^{M} \mathbb{1}(\mathbf{x}^{\mathbf{m}} \neq \tilde{\mathbf{x}}^{\mathbf{m}}) \quad (4)$$

We not only want to change a minimum number of features, but also want those features to be *informative*. For example, the explanation "Had the frequency of '!' and '!!!' is more than 0.3,

the email would be classified as spam rather than ham" is not as informative as "Had the frequency of '!' and 'wonder' is more than 0.3, the email would be classified as spam rather than ham." Hence, we want $\mathcal{S}$ to contain a list of perturbed features such that any pairwise mutual information among them is within an upper-bound γ. Thus, we add the constraint:

$$SU(X^i, X^j) \leq \gamma \quad \forall i, j \in \mathcal{S} \quad (5)$$

where $SU(\cdot)$ is *Symmetrical Uncertainty* function, a normalized form of mutual information, to be introduced in Section 4.2. Finally, we also need to ensure that the final predicate $\mathcal{P}$ is realistic. For example, the age feature should be a positive integer). Therefore, we want to generate $\tilde{x}$ such that it conforms to features domain constraints of the dataset:

$$\tilde{x} \in dom(X) \quad (6)$$

Newly introduced constraints are novel from previous adversarial literature, which focus more on minimizing the difference $\|\boldsymbol{x} - \tilde{x}\|_p$. However, minimizing such a distance alone will not necessarily make $\tilde{x}$ more self-explanatory to users. Instead, we propose that as long as the constraints on the maximum number of perturbed features, i.e., Eq. (3), their entropy, i.e., Eq. (5), and domain, i.e., Eq. (6), is satisfied, we can generate more concise and informative explainable contrastive samples. From the above analysis, we formalize the objective function as follows.

OBJECTIVE FUNCTION: Given $\boldsymbol{x}$, hyperparameter K, γ, our goal is to generate new contrastive sample $\tilde{x}$ to explain the prediction $f(\boldsymbol{x})$ by solving the objective function:

$$\min_{\tilde{\boldsymbol{x}}} \quad dist(\tilde{\boldsymbol{x}}, \boldsymbol{x})$$

$$s.t. \quad argmax(f(\boldsymbol{x})) \neq argmax(f(\tilde{\boldsymbol{x}})), \quad |\mathcal{S}| \leq K \quad (7)$$

$$SU(X^i, X^j) \leq \gamma \quad \forall i, j \in \mathcal{S}, \quad \tilde{x} \in dom(X)$$

4 GRACE: GENERATING INTERVENTIVE CONTRASTIVE SAMPLES FOR MODEL EXPLANATION

This section describes how to solve the objective function and the details of GRACE. Figure 1 gives an illustration of the framework. It consists of three steps: (i) entropy-based forward features ranking, which aims at finding instance-dependent features satisfying the constraint; (ii) generate contrastive samples with the selected features; and (iii) create an explanation text based on generated sample $\tilde{\boldsymbol{x}}$. Alg. 1 describes GRACE algorithm.

4.1 Contrastive Sample Generation Algorithm

Before introducing how to obtain a list of potential features to perturb, in this section, we first describe our contrastive sample generation algorithm by assuming that the ordered feature list $\mathcal{U}^*$ is given. To solve $\tilde{\boldsymbol{x}}$ such that Eq. (2) is satisfied, we can continuously perturb $\tilde{\boldsymbol{x}}$ by projecting itself on the decision hyperplane separating it with the nearest contrastive class v. Particularly, at each time-step i, we project $\tilde{\boldsymbol{x}}$ with orthogonal projection vector $\mathbf{r}_v$:

$$\mathbf{r}_v = \frac{|f_v(\tilde{\boldsymbol{x}}_{i-1}) - f_C(\boldsymbol{x})|}{\|\nabla f_v(\tilde{\boldsymbol{x}}_{i-1}) - \nabla f_C(\boldsymbol{x})\|_2^2} \left(\nabla f_v(\tilde{\boldsymbol{x}}_{i-1}) - \nabla f_C(\boldsymbol{x}) \right) \quad (8)$$

where $f_v(\tilde{\boldsymbol{x}}_{i-1})$ is the confidence of f on $\tilde{\boldsymbol{x}}_{i-1}$ being classified as class v. $C \longleftarrow argmax(f(\boldsymbol{x}))$ is the current prediction label, and contrastive class $\boldsymbol{v}$ can be inferred with Alg. 2, Ln. 2. Intuitively, $\boldsymbol{v}$

Algorithm 1 GRACE

Input: $f, \boldsymbol{x}, K, \gamma, \mathcal{X}$
Output: $\tilde{\boldsymbol{x}}, \tilde{y}$

1: *Initialize:* $\tilde{\boldsymbol{x}} \longleftarrow \boldsymbol{x}, j \longleftarrow 1$
2: $\mathcal{U} \longleftarrow W_{Gradient}(f, \boldsymbol{x})$ **OR** $W_{Local}(f, \boldsymbol{x})$
3: $\mathcal{U}^* \longleftarrow W_{Entropy}(\mathcal{X}, \mathcal{U}, \gamma)$
4: **while** $\operatorname{argmax}(f(\tilde{\boldsymbol{x}})) = \operatorname{argmax}(f(\boldsymbol{x}))$ **AND** $k \le K$ **do**
5:    $\tilde{\boldsymbol{x}} \longleftarrow$ GENERATECONTRASTIVESAMPLE$(f, \boldsymbol{x}, \mathcal{U}^*[: k])$
6:    $k \longleftarrow k + 1$
7: $\tilde{y} \longleftarrow \operatorname{argmax}(f(\tilde{\boldsymbol{x}}))$
8: **Return** $\tilde{\boldsymbol{x}}, \tilde{y}$

x = [0.2, 0.3,..0.4, 0.1]
x* = [0.2, 0.3,..0.0, 0.1]

The email is classified as "Spam" **RATHER THAN** "Ham" **BECAUSE** there is more "!" in the message

Had frequency of "!" decreased 40%, the email would have been classified as "Ham" **RATHER THAN** "Spam"
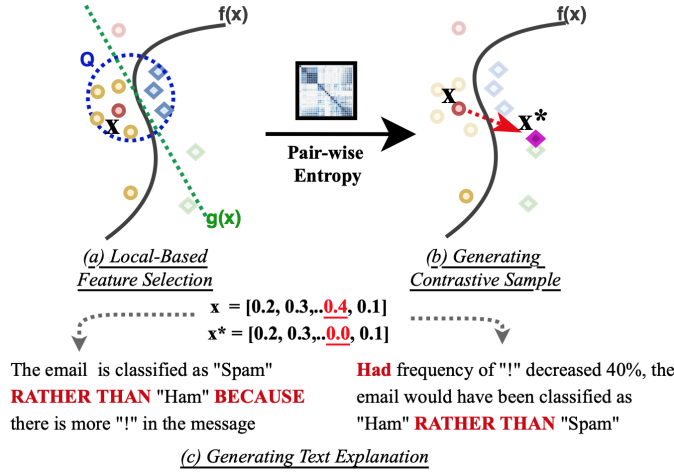
(c) Generating Text Explanation

Figure 1: GRACE with LOCAL-BASED Feature Ranking is the contrastive class across the *closest* hyperplane of the decision boundary from $\boldsymbol{x}$.

Algorithm 2 GENERATECONTRASTIVESAMPLE

Input: $f, \boldsymbol{x}, \mathcal{S}$
Output: $\tilde{\boldsymbol{x}}$

1: *Initialize:* $\tilde{\boldsymbol{x}}_0 \longleftarrow \boldsymbol{x}, i \longleftarrow 0, C \longleftarrow \operatorname{argmax}(f(\boldsymbol{x}))$
2: $v \longleftarrow \operatorname{argmin}_{c \ne C} \frac{|f_c(\boldsymbol{x}) - f_C(\boldsymbol{x})|}{||\nabla f_c(\boldsymbol{x}) - \nabla f_C(\boldsymbol{x})||_2}$
3: **while** $\operatorname{argmax}(f(\tilde{\boldsymbol{x}}_i)) = \operatorname{argmax}(f(\boldsymbol{x}))$ **AND** $i <$ STEPS **do**
4:    $\mathbf{r}_v = \frac{|f_v(\tilde{\boldsymbol{x}}_{i-1}) - f_C(\boldsymbol{x})|}{||\nabla f_v(\tilde{\boldsymbol{x}}_{i-1}) - \nabla f_C(\boldsymbol{x})||_2^2} (\nabla f_v(\tilde{\boldsymbol{x}}_{i-1}) - \nabla f_C(\boldsymbol{x}))$
5:    $\tilde{\boldsymbol{x}}_{i+1}[\mathcal{S}] \longleftarrow \tilde{\boldsymbol{x}}_i[\mathcal{S}] + \boldsymbol{r}_v[\mathcal{S}]$
6:    $\tilde{\boldsymbol{x}}_{i+1} \longleftarrow P(\tilde{\boldsymbol{x}}_{i+1}, dom(\mathcal{X}))$
7:    $i \longleftarrow i + 1$
8: **Return** $\tilde{\boldsymbol{x}}$

To address constraint Eq. (3), instead of perturbing all of the features, we update $\tilde{\boldsymbol{x}}$ only on the first $k \le K$ features from an ordered list $\mathcal{U}^*$, which will be introduced later, at each time step i until it crosses the decision boundary:

$$\mathcal{S} \longleftarrow \mathcal{U}^*[: k], \quad \tilde{\boldsymbol{x}}_i[\mathcal{S}] \longleftarrow \tilde{\boldsymbol{x}}_{i-1}[\mathcal{S}] + \mathbf{r}_v[\mathcal{S}] \quad (9)$$

Since feature perturbation based on $\mathbf{r}_v$ does not always guarantee that resulted $\tilde{\boldsymbol{x}}_i$ still maintains in the original feature space, to address constraint Eq. (6), we project those adjusted features back on to the original domain of $\mathcal{X}$:

$$\tilde{\boldsymbol{x}}_i \longleftarrow P(\tilde{\boldsymbol{x}}_i, dom(\mathcal{X})) \quad (10)$$

where P is a projection which ensures that final $\tilde{x}$ looks more real (e.g., age feature should be a whole number and > 0). The domain space $dom(\mathcal{X})$ can include the maximum, minimum, and data types (e.g., int, float, etc.) of each feature. These can be either calculated from the original training set or manually set by domain experts.

With a fixed k, Eq. (10) does not guarantee that $\tilde{x}$ will always cross the decision boundary to class $\boldsymbol{v}$. Hence, we gradually increase $k \longrightarrow K$ until a contrastive sample is successfully generated, i.e., $\operatorname{argmax}(f(\boldsymbol{x})) \ne \operatorname{argmax}(f(\tilde{\boldsymbol{x}}_i))$ or when $k == K$. Alg. 2 illustrates the steps to generate contrastive samples.

One obvious challenge is how to come up with the ordered list $\mathcal{U}^*$ of features to perturb. Next, we will describe this in detail.

4.2 Entropy-Based Forward Feature Ranking

As different $\boldsymbol{x}$ might require a different subset of features to perturb, the first challenge is how to prioritize features that are highly vulnerable to the contrastive class $\boldsymbol{v}$. To do this, we rank all features of $\boldsymbol{x}$ according to their predictive power w.r.t prediction $f(\boldsymbol{x})$, resulting in an ordered list $\mathcal{U}$:

$$\mathcal{U} \longleftarrow W(f, \boldsymbol{x}) \quad (11)$$

where $W(\cdot)$ is a feature ranking function. The most straight forward way is to rank all features according to their gradients w.r.t the *nearest* contrastive class $\boldsymbol{v}$ that back-propagates through $f(\boldsymbol{x})$, resulting in $W_{\text{GRADIENT}}(f, \boldsymbol{x})$ that returns the ranking of the following set:

$$\{\nabla f_{\boldsymbol{v}}(\boldsymbol{x}^1), \nabla f_{\boldsymbol{v}}(\boldsymbol{x}^2), .. \nabla f_{\boldsymbol{v}}(\boldsymbol{x}^M)\} \quad (12)$$

While this method is straightforward, these gradients capture a global view of feature rankings, rather than being customized to a local vicinity of decision boundary around $\boldsymbol{x}$. To overcome this limitation, we introduce $W_{\text{LOCAL}}(f, \boldsymbol{x})$ to return the ranking of the following set:

$$\{w_{g(\boldsymbol{x})}^1, w_{g(\boldsymbol{x})}^2, ... w_{g(\boldsymbol{x})}^M\} \quad (13)$$

with $w_{g(\boldsymbol{x})}^j$ is the feature importance score of the j-th feature returned from an interpretable ML model $g(\boldsymbol{x})$ (e.g., feature weights for logistic regression, Gini-score for decision tree, etc.). $g(\cdot)$ is trained on a subset of data points Q surrounding $\boldsymbol{x}$ (Figure 1a) with *maximum likelihood estimation (MLE)* as the loss function:

$$\min_{\theta_{g(x)}} \frac{1}{|Q|} \sum_{\boldsymbol{x} \in Q} f(\boldsymbol{x}) \log(g(\boldsymbol{x})) \quad (14)$$

If the prediction of $g(\boldsymbol{x})$ on Q is very close to that of $f(\boldsymbol{x})$, important features from $g(\boldsymbol{x})$ are more prone to change in $f(\boldsymbol{x})$. Q can be collected by sampling q nearest data points to $\boldsymbol{x}$ from *each of the predicted classes* by $f(\boldsymbol{x})$ on the training set using *NearestNeighbors(NN)* search algorithm with different distance functions. We set $q = 4$ and use *Euclidean distance* throughout all experiments.

In the aforementioned variants of $W(\cdot)$, each feature is treated independently with each other. However, if a pair of selected features are highly dependent on each other (e.g., frequency of "!" and "!!"), the final generated samples will be less informative. Because of this, also to address constraint Eq.(5), we want to generate a new ordered list $\mathcal{U}^*$ as follows:

$$\mathcal{U}^* \longleftarrow W_{Entropy}(\mathcal{X}, \mathcal{U}) \quad (15)$$

where $W_{Entropy}$ is a forward-based selection approach, which will *iteratively* add each feature from $\mathcal{U}$ (from the most to least

Table 2: Examples of generated contrastive samples and their explanation texts

Dataset	Features/*Prediction*	Type	Original	Generated	Changes	Explanation Text
cancer95	bare_nuclei	int	1	10	↑ 9	"if there were 9 more bare nucleus, the patient would be classified as
	diagnosis		benign	malignant		malignant *RATHER THAN* benign"
spam	word_freq_credit	float	0.470	0.225	↓ 0.245	"The message is classified as spam *RATHER THAN* ham because the
	word_freq_money	float	0.470	0.190	↓ 0.280	word 'credit' and 'money' is used twice as frequent as that of ham
	class		Spam	Ham		message"

predictive) to $\mathcal{U}^*$ one by one such that the mutual information of *any* pairs of features in $\mathcal{U}^*$ is within a upper-bound γ:

$$SU(\mathcal{X}^i, \mathcal{X}^j) \leq \gamma \quad \forall i, j \in \mathcal{U}^* \tag{16}$$

where $SU \in [0, 1]$ is an entropy-based *Symmetrical Uncertainty* [9] function that measures the mutual information between the i-th and j-th feature of $\mathcal{X}$ as:

$$SU(\mathcal{X}^i, \mathcal{X}^j) = 2\left[\frac{IG(\mathcal{X}^i | \mathcal{X}^j)}{H(\mathcal{X}^i) + H(\mathcal{X}^j)}\right] \tag{17}$$

where $IG(\mathcal{X}^i | \mathcal{X}^j)$ is the *information gain* of $\mathcal{X}^i$ given $\mathcal{X}^j$, and $H(\mathcal{X}^i)$, $H(\mathcal{X}^j)$ is empirical entropy of $\mathcal{X}^i$ and $\mathcal{X}^j$, respectively. A value $SU = 1$ indicates that $\mathcal{X}^i$ completely predicts the value of $\mathcal{X}^j$ [34]. In implementation, we first *normalize* $\mathcal{X}$ and adapt multi-interval discretization [?] to convert continuous features to discrete format before applying $SU(\cdot)$. Alg. 3 describes function $W_{Entropy}$ in details. One obvious challenge is how to come up with the ordered list $\mathcal{U}^*$ of features to perturb. The next section will describe this in detail.

Algorithm 3 Feature Selection with Entropy: $W_{Entropy}$

Input: $\mathcal{X}, \mathcal{U}, \gamma$
Output: Ordered list of features $\mathcal{U}^*$

1: *Initialize:* $\mathcal{U}^* \longleftarrow \{\}, \hat{\mathcal{X}} \longleftarrow \text{normalize}(\mathcal{X})$
2: **for** i in $\mathcal{U}$ **do**
3:    $to\_add \longleftarrow True$
4:    **for** j in $\mathcal{U}^*$ **do**
5:       **if** $SU(\hat{\mathcal{X}}^i, \hat{\mathcal{X}}^j) > \gamma$ **then** $to\_add \longleftarrow False$
6:    **if** $to\_add = True$ **then** $\mathcal{U}^* \longleftarrow \mathcal{U}^* \cup \{i\}$
7: **Return** $\mathcal{U}^*$

4.3 Generating Explanation Text

After generating contrastive sample $\tilde{\mathbf{x}}$, we take a further step and generate an explanation in natural text. Table 2 shows generated contrastive samples and the corresponding explanation for various datasets with $K = 5$. To do this, for a specific prediction $f(\mathbf{x})$ and generated contrastive sample $\tilde{\mathbf{x}}$, we first calculate their feature differences, resulting in predicate $\mathcal{P}$ as defined in Def. 1. Then, we can translate $\mathcal{P}$ to text by using condition-based text templates such as `... is classified as X RATHER THAN Y because ...`, or `had..., it would have been classified as X RATHER THAN Y` (Figure 1c). Different text templates can be selected randomly to induce diversity in explanation text. The difference in features values can be described in three different degrees of obscurity from (i) *extract value* (e.g., `0.007 point lower`), to (ii) *magnitude comparison* (e.g., `twice as frequent`), or (iii) *relative comparison* (e.g., `higher, lower`). Which degree of detail to best use is highly dependent on the specific feature, domain, and the choice of end-users, and they do not need to be consistent among perturbed features in a single explanation text.

Table 3: Dataset statistics and prediction performance

Size	Dataset	#Class	#Feat.	#Data	$Acc.^*$	$F1^*$
small	eegeye	2	14	14980	0.858	0.858
	diabetes	2	8	768	0.779	0.777
	cancer95	2	9	699	0.963	0.963
	phoneme	2	5	5404	0.774	0.772
	segment	7	19	2310	0.836	0.817
	magic	2	10	19020	0.862	0.859
medium	biodeg	2	41	1055	0.853	0.851
	spam	2	57	4601	0.932	0.932
	cancer92	2	30	569	0.958	0.958
large	mfeat	10	216	2000	0.943	0.936
	musk	2	166	476	0.783	0.789

(*) Accuracy and F1 scores are averaged across 10 different runs.

4.4 Complexity Analysis

According to Alg. 1, we analyze the time complexity of GRACE on each prediction instance as follows. The predictive feature ranking step using $\mathcal{W}_{Gradient}$ takes $O(M \log M)$ with *Quick Sort*. Reordering the ranked list of features with $W_{Entropy}$ takes $O(M)$. Generating contrastive sample step takes $O(M) + Z\mathcal{V}$ with $K \ll M$, where Z is total number of classes to predict, and $\mathcal{V}$ is the time complexity of forward and backward pass of $f(\mathbf{x})$. Generating an explanation text then takes another $O(M)$. To sum up, the overall time complexity of GRACE to generate an explanation for each prediction instance is $O(M \log M) + Z\mathcal{V}$ with $K \ll M$ (excluding the overhead of training $g(\cdot)$ and searching for Q in case of W_{Local}).

5 EXPERIMENTS

In this section, we conduct experiments to verify the effectiveness of GRACE. Specifically, we want to answer two questions: (i) how good are the generated interventive contrastive samples? and (ii) how good are the generated explanation?

5.1 Datasets

We select 11 publicly available datasets of different domains and scales from [7] to fully evaluate and understand how well GRACE works with neural networks trained on data with varied properties. As shown in Table.3, the datasets are grouped into three scale levels according to the number of features. Each dataset is split into training, validation, and test set with a ratio of 8:1:1, respectively. The table also includes the performance of different neural models (Appendix A.2.1) on test set in both Accuracy and F1 score.

5.2 Compared Methods

Since our proposed framework combines the best of both worlds: adversarial generation and neural network model explanation, we select various relevant baselines from two aspects.

- NEARESTCT: Instead of generating synthetic contrastive sample for explanation for data point $\mathbf{x}$, this approach selects the

Table 4: All results are averaged across 10 different runs. The best and second best results are highlighted in bold and underline.

Statistics	Dataset	# Features < 30						30 ≤ # Features < 100			100 ≤ # Features	
		eegeye	diabetes	cancer95	phoneme	segment	magic	biodeg	spam	cancer92	mfeat	musk
$R_{avg\#Feats}$	NearestCT	13.56	6.93	5.92	4.82	16.10	9.97	20.53	17.50	29.97	204.22	147.86
	DeepFool	14.00	8.00	9.00	5.00	19.00	10.00	41.00	57.00	30.00	216.00	166.00
	GRACE-Local	<u>1.15</u>	**1.55**	<u>2.7</u>	**1.25**	**2.42**	<u>1.68</u>	<u>3.07</u>	<u>2.95</u>	**3.95**	<u>3.28</u>	<u>3.74</u>
	GRACE-Gradient	**1.0**	<u>1.96</u>	**2.66**	<u>1.3</u>	<u>3.84</u>	**1.6**	**1.93**	**1.09**	<u>4.5</u>	**2.76**	**2.85**
$R^*_{info-gain}$	NearestCT	<u>0.69</u>	0.44	**0.64**	0.12	0.19	0.04	0.44	<u>0.62</u>	0.02	<u>0.58</u>	0.28
	DeepFool	**0.7**	0.41	<u>0.62</u>	0.12	<u>0.33</u>	0.05	<u>0.58</u>	0.53	0.01	**0.59**	0.29
	GRACE-Local	0.64	**0.79**	0.49	**0.81**	**0.55**	<u>0.67</u>	0.46	0.47	**0.13**	0.34	<u>0.3</u>
	GRACE-Gradient	0.64	<u>0.62</u>	0.52	<u>0.78</u>	0.23	**0.71**	**0.76**	**0.95**	<u>0.04</u>	0.50	**0.4**
$R_{influence}$	NearestCT	0.05	0.06	0.11	0.03	0.01	0.00	0.02	0.04	0.00	0.00	0.00
	DeepFool	0.05	0.05	0.07	0.02	0.02	0.00	0.01	0.01	0.00	0.00	0.00
	GRACE-Local	<u>0.55</u>	**0.52**	<u>0.18</u>	**0.65**	**0.23**	<u>0.4</u>	<u>0.15</u>	<u>0.16</u>	**0.04**	<u>0.1</u>	<u>0.08</u>
	GRACE-Gradient	**0.64**	<u>0.33</u>	**0.2**	<u>0.61</u>	<u>0.06</u>	**0.45**	**0.4**	**0.88**	<u>0.01</u>	**0.18**	**0.14**

nearest contrastive samples of x from the *training set* to provide contrastive explanation for the prediction $f(x)$.

- DeepFool[22]: An effective approach that was originally proposed for untargeted attack by generating adversarial samples. Even though DeepFool is not designed for generating samples to explain predictions, we consider this as a baseline that intervenes on *all features* to generate contrastive samples.
- Lime [25]: A *local interpretable model-agnostic explanation* approach that provides explanation for individual prediction. This approach replies on visualization of feature importance scores (for text and tabular data), and feature heat-map (for image data) to deliver explanation. We use an out-of-the-box toolkit[1] to run experiments for comparison. Lime is selected mainly due to its popularity as a baseline for ML explanation approach.
- GRACE-Gradient and GRACE-Local (**ours**): GRACE with predictive feature ranking function W as $W_{Gradient}$ and W_{Local}, respectively.

5.3 Evaluation of Generated Samples

In this section, we want to examine the quality of generated contrastive samples. Since DeepFool, NearestCT and GRACE generate intermediate samples to explain predictions, while Lime is not, we compare and analyze Lime separately in Section 5.4 to evaluate final generated explanation.

For each dataset, we train a neural network model $f(\cdot)$ using the training set. We tune it using the validation set together with early-stopping strategy to prevent overfitting and report its performance on the test set. Table 3 reports the averaged performance across 10 different runs. We set $K = 5$, $\gamma = 0.5$, and generate $\tilde{x}$ to explain predictions $f(x)$ of all samples in test set, resulting in the set of generated contrastive samples $\tilde{X}$.

To thoroughly examine the proposed approach, we come up with the following analytical questions (AQs).

AQ1 Fidelity: How accurate are the generated contrastive samples' labels, i.e., whether they can cross neural network model's decision boundary as expected?

AQ2 Conciseness: How concise are generated samples, i.e., how many features needed to be perturbed to successfully generate contrastive samples?

AQ3 Info-gain: How informative are generated samples?

AQ4 Influence: Derived from Def. 2, how well do the generated samples answer the question *Why X rather than Y?*.

5.3.1 AQ1 (Fidelity) Fidelity, measured by $R_{fidelity}$, shows how accurately contrastive samples are generated according to the neural network model's boundary, i.e., the accuracy of generated samples' labels w.r.t their predictions by the neural network model:

$$R_{fidelity} = \frac{1}{|\tilde{X}|} \sum_{(\tilde{x}, \tilde{y}) \in \tilde{X}} \mathbb{1}(\tilde{y} = \text{argmax}(f(\tilde{x}))) \quad (18)$$

Different from the two baselines, two variants of GRACE have to satisfy the domain constraints, minimize the number of features, and their entropy, all at the same time. Nevertheless, with $K = 5$, our method shows an average $R_{fidelity}$ of around 0.8 for both GRACE-Gradient and GRACE-Local. As the # of perturbed features increases, the Fidelity scores for both GRACE-Gradient and GRACE-Local also increase, which satisfies the expectation (Figure 2).
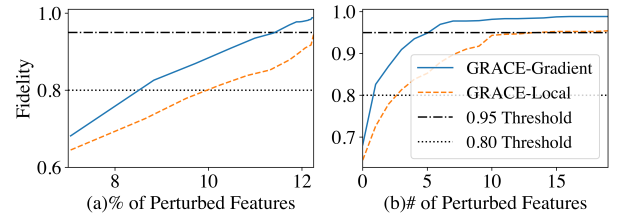

Figure 2: Percentage of perturbed features v.s fidelity

5.3.2 AQ2 (Conciseness) We not only want to generate samples with high *fidelity*, but also want to perturb as few features as possible. Thus, we introduce *conciseness* that measures the ability to generate $\tilde{x}$ by changing as few features as possible. To do this, we want to see how *fidelity* correlates with the average number of perturbed features, denoted as $R_{avg\#Feats}$:

$$R_{avg\#Feats} = \frac{1}{|\tilde{X}|} \sum_{\tilde{x} \in \tilde{X}} |S_{\tilde{x}}| \quad (19)$$

where $S_{\tilde{x}}$ returns the list of features to perturb in x to generate $\tilde{x}$.

Table 4 shows that our GRACE framework dominates the two baselines DeepFool and NearestCT on $R_{avg\#Feats}$ by large margin. Specifically, with $K = 5$, our approach is able to generate contrastive samples with much less number of perturbed features,

averaging around less than 2.5 features across all datasets. Interestingly, GRACE-Gradient was able to change on average less than 3 out of a total of 216 and 166 features in the case of *mfeat* and *musk* dataset, respectively.

Moreover, while our method only needs to use as few as 12.5% of total # of features to achieve fidelity of around 0.95 (Figure 2), DeepFool and NearestCT baseline needs to change almost 100% of the total # of features to achieve a similar score.

5.3.3 AQ3 (Info-gain)
Since we want to generate samples that are informative, we hope to minimize the averaged mutual information of all pairs of selected features across all samples in $\tilde{\mathcal{X}}$. Hence, we formulate $\mathbf{R}_{\text{info-gain}}$ to measure such characteristic of being informative as follows:

$$\mathbf{R}_{\text{info-gain}} = 1 - \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}} \sum_{i \in \mathcal{S}_{\tilde{\mathbf{x}}}} \sum_{j \in \mathcal{S}_{\tilde{\mathbf{x}}}} \frac{\text{SU}(\mathcal{X}^i, \mathcal{X}^j)}{|\mathcal{S}_{\tilde{\mathbf{x}}}|^2} \quad (20)$$

To be fair with other baselines, we instead report $\mathbf{R}^*_{\text{info-gain}} = \mathbf{R}_{\text{info-gain}} \times \mathbf{R}_{\text{fidelity}}$ to take into account the fidelity score. Even so, thanks to the entropy-aware feature selection mechanism, GRACE is able to generate contrastive samples that are much more informative compared to DeepFool's in most of the datasets (Table 4). This shows that samples generated by our framework are not only contrastive but also informative to the final explanation.

5.3.4 AQ4 (Influence)
Extended from influence score with tolerance parameter $\lambda = 1$ (Def. 2), we aim to measure how well the generated samples can influence the explanation of a specific prediction. Denoted by $\mathbf{R}_{\text{influence}}$, the influence score first captures whether generated samples are still within the original domain space, or $\mathbf{R}_{\text{domain}}$ as follows:

$$\mathbf{R}_{\text{domain}} = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}} \mathbb{1}(\tilde{\mathbf{x}} \in \text{dom}(\mathcal{X})) \quad (21)$$

Moreover, the influence score is also proportional to how faithful generated samples are to the neural network model's decision boundary ($\mathbf{R}_{\text{fidelity}}$), how informative they are ($\mathbf{R}_{\text{info-gain}}$), how concise in terms of number of perturbed features ($\mathbf{R}_{\text{avg\#Feats}}$), resulting in a $\mathbf{R}_{\text{influence}}$ calculated as follows:

$$\mathbf{R}_{\text{influence}} = \frac{\mathbf{R}_{\text{fidelity}} \times \mathbf{R}_{\text{info-gain}} \times \mathbf{R}_{\text{domain}}}{\mathbf{R}_{\text{avg\#Feats}}} \quad (22)$$

Intuitively, $\mathbf{R}_{\text{influence}}$ describes the capability to generate new contrastive samples that are both informative, concise, and valid within the original domain space. Hence, the larger the score, the better.

Regarding $\mathbf{R}_{\text{domain}}$, table 4 shows that DeepFool performs worst on $\mathbf{R}_{\text{domain}}$, averaged about 0.86, since the generation might move $\tilde{\mathbf{x}}$ much further away from the original distribution, while other methods always ensure that generated samples are within the original domain space. As regards as $\mathbf{R}_{\text{influence}}$, GRACE is able to generate highly more influential contrastive samples than DeepFool and NearestCT even when taking $\mathbf{R}_{\text{fidelity}}$ into account, which is the strongest point of both two baselines.

5.4 Evaluation of Generated Explanation

In this section, we want to compare GRACE with Lime [25] from end-users' perspectives on their generated explanation. Before introducing user-studies to compare between two methods, we first draw some observations in a case-study below.

Table 5: User-study with hypothesis testing to compare explanation generated by GRACE against Lime

	Alternative Hypothesis	t-stats	p-value	df
$\mathcal{H}_1$	GRACE is more intuitive and friendly	2.3115	0.0104*	42
$\mathcal{H}_2$	GRACE is more comprehensible	3.0176	0.0013**	42
$\mathcal{H}_3$	GRACE leads to more accurate actions	4.4875	$3.39e^{-5}$**	37

*reject Null hypothesis with p-value<0.05 (95% CI) on one-tailed t-test
**reject Null hypothesis with p-value<0.0 1 (99% CI) on one-tailed t-test

5.4.1 Case-study: breast-cancer diagnosis :
We select *cancer95* dataset to experiment. Following the same experimental setting in Section 5, we apply Lime and GRACE on the trained neural network model to explain its predictions on the test set. Figure Sup1 depicts explanation produced by Lime on a patient diagnosed as malignant by the model. Following guideline published by Lime's author [2], explanation for each feature can be interpreted as follows: `"if bare_nuclei is less than or equal 6.0, on average, this prediction would be 0.15 less malignant"`. With the same prediction, GRACE generates an explanation text as follows: `"Had bare_nuclei been 7.0 point lower and clump_thickness been 9.0 point lower, the patient would have been diagnosed as` **`benign rather than malignant`**`"`
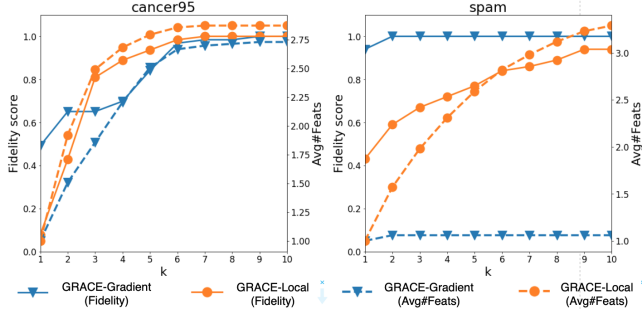
Method wise, both are *instance-based* explanation algorithms, or both explain individual predictions. From Figure Sup1, by presenting top-k important features, Lime does not convince if and how a single or combinations of features are vulnerable to the contrastive class, but this is very vivid and concise in case of GRACE. Moreover, while both methods provide some intuition on decisive thresholds at which the prediction would change its direction, the thresholds provided by Lime is *only* a local approximation, while that provided by GRACE (e.g., 7.0 point lower for bare_nuclei) is faithful to the neural network model (*fidelity score* is 1.0). Overall, the explanation generated by GRACE is more concise and faithful to the decision boundary of the neural network model.

5.4.2 User-Study 1: Intuitiveness, friendliness & comprehensibility :
We have recruited participants on Amazon Mechanical Turk (AMT) and asked them to compare two explanation methods: Lime and GRACE. Without assuming or requiring any ML background on the participants, we want to test two alternative hypothesises: explanation generated by GRACE is ($\mathcal{H}_1$) more intuitive and friendly, and ($\mathcal{H}_2$) more comprehensible than that generated by Lime to general users. To test these, using the same prediction instance, we first generate explanation text by Lime and GRACE. While by default Lime returned explanation for top 10 features, we limit only 5 features that are the most significant. On the contrary, GRACE only needs 2 features for generating contrastive explanation. Since Lime method originally does not generate explanation text, we then translate its result to text interpretation as described in previous case-study. Finally, we ask the participants to rank on a scale from 1 to 10 for each question (i) and (ii). We did the surveys for each method *separately*.

From Table 5 and Figure 4, it is significant (*p-value* ≤ 0.05) that GRACE is able to generate more intuitive, friendly ($\mathcal{H}_1$, 6.35 v.s. 4.76 in mean ranking), and more comprehensible ($\mathcal{H}_2$, 7.35 v.s. 5.52 in mean ranking) explanation than Lime for general users. We

---
[2]https://github.com/marcotcr/lime

Figure 3: Effects of K on R_{fidelity} and $R_{\text{avg\#Feats}}$ score

Figure 4: Comparison of generated explanation: GRACE v.s. Lime. **Scores are normalized to [0,1]**

Table 6: Effects of entropy threshold γ on $R_{\text{info−gain}}$

Dataset	Method	1.0	0.7	0.5	0.3
musk	GRACE-Gradient	0.51	0.51	**0.58**	**0.58**
	GRACE-Local	0.36	0.36	**0.54**	**0.54**
segment	GRACE-Gradient	0.57	0.57	**0.59**	**0.59**
	GRACE-Local	0.79	0.79	**0.84**	**0.84**

also carry out an experiment that includes a visualization design showing the top 5 features and an explanation text for the top 2 features for each method as shown in Figure Sup1. This too results in favorable results for GRACE over Lime.

5.4.3 User-Study 2: How much end-users indeed understand the explanation? In practice, ML models usually play the role of assisting human to make informative decisions [17]. Therefore, extending from previous experiment, we hypothesize that a good explanation should not only be comprehensible, but should also help materialize in accurate decision. Here we want to test workers' actual understanding from the explanation with alternative hypothesis ($\mathcal{H}_3$): users who are provided with explanation generated by GRACE are better at making post-explanation decision than those provided with explanation generated by Lime. To test this, we first present the same prediction scenario from previous user-study, then ask each participant to analyze the explanation and adjust the sample's feature values such that the model would change its prediction (e.g., from malignant to benign). This task requires the worker to recognize from the explanation hints of both (i) what the key features are and (ii) how the changes of those features affect the model's prediction. To ensure the quality of the workers, we only select workers with "US Graduate Degree" as a qualification provided by AMT. We use the trained model to validate the responses and report the average accuracy. From Table 5 and Figure 4, it is highly significant (*p-value* $\leq$ 0.01) that workers who provided with explanation generated by GRACE have more accurate answers than those provided with explanation generated by Lime ($\mathcal{H}_3$), showing 0.75 v.s 0.16 of average accuracy, respectively. In other words, explanation generated by GRACE is more effective in supporting users to make tangible decisions, such as suggesting an alternative scenarios when dealing with neural network models.

5.5 Parameter Sensitivity Analysis

5.5.1 Effects of K : One of the important factors that largely affect the explainability of GRACE is the value of parameter K, or the **maximum** number of features to change during the contrastive samples generation process. While a small K is more preferable, it would become more challenging for GRACE to ensure perturbed samples to cross the decision boundary. This will eventually hurt R_{fidelity}. Here we want to see how different K values affect the generated samples' *fidelity* and the number of perturbed features. For each dataset, we next train a neural network model and test this model with all values of K = {1, 2, 3,.. 10} and plot it against respective R_{fidelity} and $R_{\text{avg\#Feats}}$. Figure 3 reports two distinctive patterns between GRACE-Gradient and GRACE-Local: (i) both
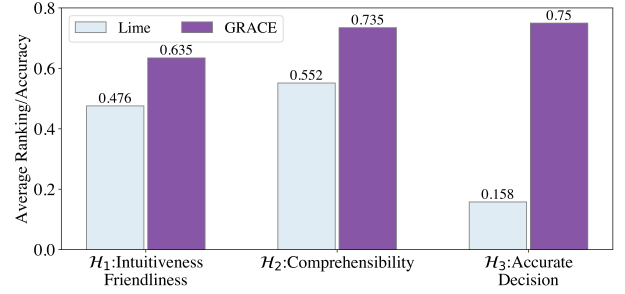
approaches witness gradual increment in R_{fidelity} and $R_{\text{avg\#Feats}}$, with neither one of them dominates the performance (e.g., *cancer95* dataset), or (ii) one of them greatly out-weights the other (e.g., *spam* dataset). Overall, by increasing K, generated samples are more faithful to the neural network model' decision boundary. Yet the average number of features needed to change to achieve so also increases, hence eventually reduce explainability.

5.5.2 Effects of entropy threshold γ : Entropy threshold γ is set to ensure that no pairs out of selected features are conveying very similar information, hence making generated samples more informative to users. Similar to the previous experiment, we keep other parameters the same while vary γ as {1.0, 0.7, 0.5, 0.3}. The results are shown in Table 6. We observed that γ is not very sensitive, showing the best value of $\gamma \leq 0.5$, which can be explained that the pair of features are usually more or less predictable given the other at a specific level. However, by setting $\gamma = 0.5$, we can observe larger improvement in case of *musk* and *segment* dataset.

6 RELATED WORK

Regarding *explanation by intervention*, our Def. 1 relates to *Quantitative Input Influence* [5], a general framework to quantify the influence of a set of inputs on the prediction outcomes. The framework follows a two-step approach: (i) it first changes each individual feature by replacing it with a random value, and *then* (ii) observes how the outcome, i.e., prediction, changes accordingly. However, we propose a more systematic way by generating a new sample at once by directly conditioning it on a contrastive outcome (X *rather than Y*). A few prior works (e.g., [19, 31, 36]) also propose to generate contrastive samples with (i) minimal corrections from its original input by minimizing the distance: $\delta = \|\boldsymbol{x} - \tilde{\mathbf{x}}\|_p$ and with (ii) minimal number of features needed to change to achieve such corrections. While Wachter et al. [31] use δ with ℓ_1 norm to induce sparsity with the hope to achieve (ii), Zhang et al. [36] approach the problem in a reverse fashion, in which they try to search for minimal δ w.r.t to a pre-defined number of features to be changed. Regardless, without considering the mutual information among pair-wise of features, it does not always guarantee that generated samples are informative to end-users. The work [30] also proposes

a method to use decision trees to search for a decisive threshold of feature's values at which the prediction will change, and utilize such threshold to generate explanations for neural network model' predictions. While sounds similar to our approach, this method shares a similar dis-merit with LIME [25] since the generated explanation is only an approximation and not faithful to the model. In this paper, we take a novel approach to generate contrastive samples that are not only contrastive but also faithful to the neural network model and "informative" to end-users.

As regards as *features selection*, we employ a forward-based approach together with *Symmetrical Uncertainty (SU)* and the approximation of features importance according to the neural network model. While there are other algorithms for ranking or selecting features (e.g., submodularity [12], ℓ_1 [31], tree-based ([30], etc.), our proposed method is selected because of it is both effective (high fidelity and informative scores as a result) and easy to implement, not to mention that *SU* can work with continuous features, and it also considers the bias effects in which one feature might have many diverse values than the other [34].

7 CONCLUSION AND FUTURE WORK

In this paper, we borrow "contrastive explanation" and "explanation by intervention" concepts from previous literature and develop a generative-based approach to explain neural network models' predictions. We introduce GRACE, a novel instance-based algorithm that provides end-users with simple natural text explaining neural network models' predictions in a contrastive *"Why X rather than Y"* fashion. To facilitate such an explanation, GRACE extends adversarial perturbation literature with various conditions and constraints, and generates contrastive samples that are concise, informative and faithful to the neural network model's specific prediction. User-studies and quantitative experiments on several datasets of varied scales and domains have demonstrated the effectiveness of the proposed approach. There are several Interesting future directions. First, in this paper, we intervene a selected subset of features without considering conditional dependencies among all variables after such intervention. This might create undesirable samples that are unrealistic (e.g., "'a pregnant man"). Thus, we plan to address interactions among the features to generate samples that are *more realistic*. Second, in this work, we assume a white-box setting that we can access the gradients of the model. We want to extend GRACE for other black-box settings, gradients of which are not accessible. Since our method works exclusively for multinomial classification task, we also plan to apply it on other ML tasks such as regression, clustering, etc.

8 ACKNOWLEDGEMENT

References

[1] Sercan O Arik and Tomas Pfister. 2019. TabNet: Attentive Interpretable Tabular Learning. *arXiv preprint arXiv:1908.07442* (2019).
[2] Björn Barz and Joachim Denzler. 2019. Deep Learning on Small Datasets without Pre-Training using Cosine Loss. *arXiv preprint arXiv:1901.09054* (2019).
[3] Babak Ehteshami Bejnordi, Mitko Veta, Van Diest, et al. 2017. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Jama* 318, 22 (2017), 2199–2210.
[4] Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. 2018. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *ACM SIGKDD/KDD*. ACM, 1244–1253.
[5] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE SP*. IEEE, 598–617.
[6] Matthew F Dixon, Nicholas G Polson, and Vadim O Sokolov. [n. d.]. Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *Applied Stochastic Models in Business and Industry* ([n. d.]).
[7] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository.
[8] Thomas Fischer and Christopher Krauss. [n. d.]. Deep learning with long short-term memory networks for financial market predictions. *EJOR* 270 ([n. d.]).
[9] Brian P Flannery, Saul A Teukolsky, William H Press, and William T Vetterling. 1988. *Numerical recipes in C: The art of scientific computing*. Vol. 2.
[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[11] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).
[12] Rajiv Khanna, Ethan Elenberg, Alexandros G Dimakis, Sahand Negahban, and Joydeep Ghosh. 2017. Scalable greedy feature selection via weak submodularity. *arXiv preprint arXiv:1703.02723* (2017).
[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[14] Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *National Academy of Sciences* 110, 15 (2013), 5802–5805.
[15] David Lewis. 2013. *Counterfactuals*. John Wiley & Sons.
[16] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066* (2015).
[17] Zachary C Lipton. [n. d.]. The mythos of model interpretability. *Queue* ([n. d.]).
[18] Samaneh Mahdavifar and Ali A Ghorbani. 2019. Application of deep learning to cybersecurity: A survey. (2019).
[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. [n. d.]. Least squares generative adversarial networks. In *CVPR*.
[20] Jan André Marais. 2019. *Deep learning for tabular data: an exploratory study*. Ph.D. Dissertation. Stellenbosch: Stellenbosch University.
[21] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. 2014. Causality and explanations in databases. *VLDB Endowment* 7, 13 (2014), 1715–1716.
[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE CVPR*. 2574–2582.
[23] Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*.
[24] Daniele Ravì, Charence Wong, Fani Deligianni, Melissa Berthelot, Javier Andreu-Perez, Benny Lo, and Guang-Zhong Yang. 2016. Deep learning for health informatics. *IEEE journal of biomedical and health informatics* 21, 1 (2016), 4–21.
[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. [n. d.]. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*.
[26] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *ACM SIGMOD*. ACM, 1579–1590.
[27] Ira Shavitt and Eran Segal. 2018. Regularization learning networks: deep learning for tabular datasets. In *NIPS*. 1379–1389.
[28] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. 2000. Scalable techniques for mining causal structures. (2000).
[29] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
[30] Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. 2018. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470* (2018).
[31] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.
[32] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment* 6, 8 (2013), 553–564.
[33] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural network-based graph embedding for cross-platform binary code similarity detection. In *ACM SIGSAC CCS*. 363–376.
[34] Lei Yu and Huan Liu. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*. 856–863.
[35] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV*. Springer, 818–833.
[36] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. 2018. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *NIPS*.
[37] Daniel John Zizzo, Daniel Sgroi, et al. 2000. *Bounded-rational behavior by neural networks in normal form games*. Nuffield College.

Table Sup1: The details of configuration and training parameters of neural network models on different datasets

Parameter	eegeye	diabete	cancer95	phoneme	segment	magic	biodeg	spam	cancer92	mfeat	musk
Size of Hidden Layers	[40,30]	[15, 7]	[15, 15]	[20, 5]	[30, 10]	[35, 20]	[60, 50]	[50, 30]	[50, 20]	[100, 50]	[100, 100]
Batch Size	512	512	512	512	512	512	512	512	512	512	512
Learning Rate	0.01	0.01	0.001	0.001	0.01	0.001	0.01	0.001	0.001	0.001	0.0001
Early-Stopping Patience	5	3	3	3	3	4	3	3	3	5	3
Maximum Epochs	500	500	500	500	500	500	500	500	500	500	500

Table Sup2: Prediction scenario from *cancer95* dataset used in case-study and user-studies

Feature	Bare_Nuclei	BlaChr	MarAdh	CluThic	Mitoses	CelSizUni	CelShaUni	NorNuc	SinEpCeSi	**Model Prediction**
Patient	10.0	4.0	5.0	8.0	1.0	5.0	5.0	3.0	2.0	**88% Malignant**

Table Sup3: The details of parameters of GRACE

Parameter	Value
K	5
γ	0.5
STEPS	200
Nearest Neighbor parameter (n_neighbors)	4
Nearest Neighbor parameter (distance function)	auto

A APPENDIX ON REPRODUCIBILITY

In this section, we provide the details of experimental configuration and the designs of our user-studies to facilitate the reproducibility of our work.

A.1 Source Code, Software, and Dataset

Software wise, we use *Python* (version 3.7.3) as the main programming language, *Scikit-learn* (version 0.21.3) and *PyTorch* (version 1.4.0) as the main machine learning frameworks. All eleven datasets used in this paper are publicly available in the UCI Machine Learning Repository [7].

A.2 Evaluation of Generated Samples

In Section 5, we compare GRACE with DeepFool and NearestCT baseline on the quality of generated contrastive samples to explain predictions of neural network model $f(\cdot)$ with parameters and configuration as follows.

A.2.1 Neural Network Model $f(\cdot)$ We employ two hidden layers fully-connected-networks with *ReLu* activation function, followed by a softmax layer to train a neural network model for each dataset with parameters and configuration described in Table Sup1. We initialize all the weight tensors and biases of the model using *Glorot initialization* [10] and value of 0.01, respectively. We use *Adam* optimizer [13] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 8$ to train the model with *Cross Entropy Loss*.

A.2.2 GRACE We follow the generation algorithm as described in Section 4, particularly Alg. 1, Alg. 2, and Alg. 3. Implementation of *Symmetrical Uncertainty (SU)* function is retrieved from public repository for *Fast Correlation-Based Filter Feature Selection*: https://github.com/shiralkarprashant/FCBF. We use Logistic Regression as the explainable ML classifier $g(\cdot)$ used in $W_{Gradient}$ function described in Section 4. We set all parameters following Table. Sup3. The descriptions of major parameters are as follows.

(1) K: Maximum number of features to perturb (as in Alg. 1)
(2) γ: Entropy upper-bound threshold (as in Alg. 3)

(3) STEPS: Maximum number of steps to project $\tilde{x}$ on the contrastive class: (as in Alg. 2)

A.2.3 DeepFool Since DeepFool is originally developed for image data, we adapt a publicly available repository at https://github.com/LTS4/DeepFool to our tabular datasets.

A.3 Evaluation of Explanation

We compare our proposed framework, GRACE, with Lime on the quality and effectiveness of generated explanation. We randomly select a sample from *cancer95* dataset and study its prediction throughout Section 5.4. Its feature values and prediction is shown in Table. Sup2.

A.3.1 Case-study: breast-cancer diagnosis : We study the prediction scenario of patient described in Table. Sup2. Figure Sup1 shows the explanation generated by Lime and GRACE for the scenario. Even though GRACE focuses on generating explanation in natural text and visualization is not our main contribution, we attach a potential visualization design corresponding to the generated explanation text as seen in Figure Sup1.

A.3.2 User-Study 1: Intuitiveness, friendliness & comprehensibility : We first present the AMT workers the prediction scenario as in Table. Sup2. To ensure the quality of the workers, we only recruit workers with the approval rate of assignments greater than 95%. To ensure the quality of the responses, we filter out ones that spend less than 1.5 minutes. Eventually, we have 37 workers participated with a total of 44 valid responses, 23 and 21 of which are asked to assess GRACE's, and Lime's explanation, respectively. An example of the task interface is depicted in Figure Sup2.

A.3.3 User-Study 2: How much end-users indeed understand the explanation? We also present the AMT workers the prediction scenario as in Table. Sup2. To ensure the quality of the workers, we apply two recruitment requirements provided by AMT: (i) the approval rate of assignments of any workers must be greater than 95%, (ii) the workers must have a "US Graduate Degree". To ensure the quality of the responses, we filter out ones that spend less than 3 minutes. Eventually, we have 24 workers participated with a total of 39 valid responses, 20 and 19 of which are provided with GRACE's, and Lime's explanation, respectively. An example of the task interface for GRACE is depicted in Figure Sup3.
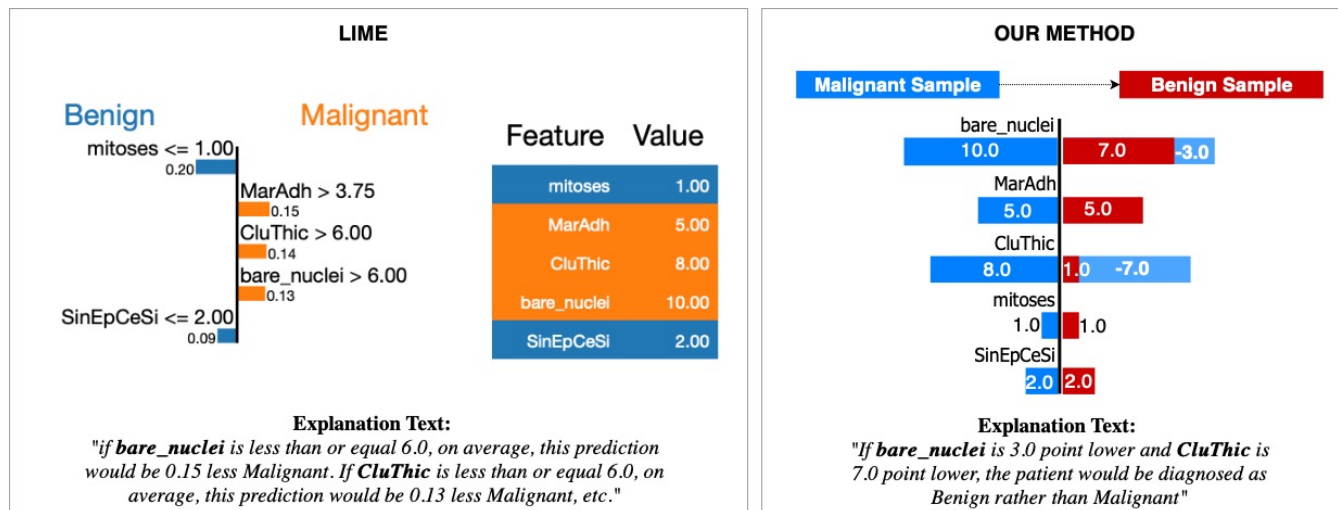
Figure Sup1: Example explanation produced by LIME and our method (GRACE).

Feature	Bare_nuclei	MarAdh	CluThic	mitoses	CelSizUni	CelShaUni	NorNuc	SinEpCeSi	Model Prediction
Value	10.0	5.0	8.0	1.0	8.0	5.0	3.0	2.0	Malignant

Explanation

*"Had **bare_nuclei** been 3.0 point lower and **CluThic** been 7.0 point lower, the patient would have been diagnosed as Benign rather than Malignant"*

Q1:Given a scale from 1 to 10, "how intuitive and friendly is the explanation to you?" (1 is least preferable, 10 is most preferable)

○——————————————— 0 ⦾

Q2:Given a scale from 1 to 10, "how understandable is the explanation to you?" (1 is least preferable, 10 is most preferable)

○——————————————— 0 ⦾

Figure Sup2: Interface of User-study 1 (GRACE: Intuitiveness, friendliness & comprehensibility).

Explanation

*"If **Bare_nuclei is 3.0 point lower** and **CluThic is 7.0 point lower**, while keeping other features the same, the patient would be diagnosed as Benign rather than Malignant"*

Q2:Below is the current value for each features of *PATIENT 1*. Following the *explanation* displayed, please *ADJUST (increase, decrease, do not change)* these values such that the computer model will change the prediction for this patient to *BENIGN*

Bare_nuclei:

●—————————————— 10 ⦾

BlaChr:

————●——————————— 4 ⦾

Figure Sup3: Interface of User-study 2 (GRACE: How much end-users indeed understand the explanation?) Note that input controls for other features (rather than *Bare_nuclei, BlaChr*) and the feature of the sample is omitted in the figure due to space issue.